J. Cryptology (1997) 10: 163–194

Journal of
CRYPTOLOGY

On the Existence of Statistically Hiding Bit Commitment Schemes and Fail-Stop Signatures*

Ivan B. Damgård,
Aarhus University, Matematisk Institut, BRiCS,**
Ny Munkegade, DK-8000 Aarhus C, Denmark
ivan@daimi.aau.dk

Torben P. Pedersen,
Cryptomathic, Klastergade 28, 1,
DK-8000 Aarhus C, Denmark
tpp@cryptomathic.dk

Birgit Pfitzmann
Universität Hildesheim, Institut für Informatik,
Marienburger Platz 22, D-31141 Hildesheim, Germany
pfitzb@informatik.uni-hildesheim.de

Communicated by Joan Feigenbaum

Received 19 April 1994 and revised 10 June 1996

Abstract. We show that the existence of a statistically hiding bit commitment scheme with noninteractive opening and public verifiability implies the existence of fail-stop signatures. Therefore such signatures can now be based on any one-way permutation. We also show that genuinely practical fail-stop signatures follow from the existence of any collision-intractable hash function. These are the weakest assumptions known to be sufficient for fail-stop signatures.

Conversely, we show that any fail-stop signature scheme with a property we call the *almost unique secret key property* can be transformed into a statistically hiding bit commitment scheme. All previously known fail-stop signature schemes have this property. We even obtain an equivalence, because we can modify the construction of fail-stop signatures from bit commitments such that it has this property.

Key words. Bit commitments, Fail-stop signatures, Digital signatures, Protocols, Zero-knowledge, Collision intractable hash functions, Privacy amplification.

* A preliminary version of this paper was presented at Crypto '93. Part of Torben Pedersen's work was done while at Aarhus Unversity, Department of Computer Science. He was supported in part by the Carlsberg Foundation. Part of Birgit Pfitzmann's work was done at the Isaac Newton Institute, University of Cambridge, England. Future address: Universität Dortmund, Informatik VI, D-44221 Dortmund 50, Germany.

** Basic Research in Computer Science, Center of the Danish National Research Foundation.

1. Introduction

1.1. *Motivation*

In this section, we introduce the two main actors on the scene, fail-stop signatures (FSS) and statistically hiding bit commitments.

Fail-stop signatures were introduced in [WP]. Further constructions appeared in [PW1], [PW2], [HP], and [HPP]. A formal definition of the concept and a survey of the recent most efficient schemes will appear in [PP].

Before going into the properties of FSS schemes, let us discuss some aspects of ordinary digital signatures: In an application of such signatures, what should happen if someone shows up with a message and a valid-looking signature from user A, but A claims that she never signed the message? Suppose the signature scheme is based on a computational problem which everybody accepts cannot be solved in polynomial time. Based on this, one could claim that it is not reasonable to assume that the system was broken by an enemy. So either A is lying, or she must have stored her secret key insecurely, and should therefore be held responsible in either case.

However, this argument sweeps under the rug a very important point: we always have to choose particular instances of the problem for each user, and the discussion should actually refer to how hard this particular instance is to break. If we are using RSA, for example, we have to decide on a size of moduli to use. Even if we believe that factoring is not in polynomial time, this does not answer questions like: "Are 512-bit moduli secure enough?" This is a question about the state of the art of practical factoring, and does not have much to do with its complexity theoretic status.

In a practical situation, it is often the case that individual users have only very limited computing power available. This limits the size of problem instance they can use, but not the amount of computing power that might be used to break those instances. In such a situation, depending on the practical circumstances, the possibility that A is not lying and someone broke her key is perhaps not so unreasonable after all.

This raises a natural question: Is it possible at all to distinguish between:

- on the one hand, the case where A is lying or has leaked the secret key; and
- on the other hand, the case where someone with a large (unexpected) amount of computing power has broken the system?

This is precisely what FSS schemes enable us to do. The crucial property that distinguishes FSS from ordinary digital signatures is that there are several possible secret keys corresponding to a given public key. Even a computationally unbounded enemy cannot guess from publicly available information which of the possible secret keys is known to the signer. As usage of different secret keys in general leads to different signatures, it is impossible for the enemy to predict which signature the signer would produce on a given message, if it has not been signed yet.

Furthermore, from two different signatures on the same message, A can produce what is known as a *proof of forgery*. But if she has only the signature available that she would produce herself, it is not feasible for her to produce such a proof. Thus if a computationally unbounded enemy tries to frame A and submits a message seemingly signed by A, with overwhelming probability the signature will not be the one A would

produce herself, and A can therefore respond with a proof of forgery. On the other hand, A cannot falsely repudiate her own signature, if it has in fact not been forged, unless she herself breaks the computational assumption. (Thus even in this case, the proof of forgery correctly indicates that someone with unexpectedly large computing power has broken the scheme.)

In this paper, we show that there is an intimate connection between statistically hiding bit commitment schemes and FSS schemes. A bit commitment scheme is a protocol that party A can conduct with B to commit herself to a bit b without revealing to B (or anyone else) the value of b. At a later time, A can open the commitment and convince B about the value that was chosen originally, i.e., it is not feasible for A to open a commitment to reveal both $b = 0$ and $b = 1$. A commitment scheme is said to be *statistically hiding* if B gets only negligible Shannon-information about b prior to the opening of the commitment. Such bit commitment schemes are extremely important, because their existence implies perfect or statistical zero-knowledge arguments for any problem in NP [BCC].

1.2. *Overview of the Results*

Concretely, we show how to construct FSS schemes from any statistically hiding bit commitment scheme with noninteractive opening and public verifiability (see below for details). This result is also contained more or less implicitly in [PW1]; it was also discussed informally, prior to the work on this paper, by Moti Yung and Birgit Pfitzmann. Our contribution in this respect is to simplify somewhat the construction and to identify the properties needed from the bit commitment scheme. By [NOVY], this means that FSS schemes can be based on any one-way permutation.

We also show that any collision-intractable hash function can be used to build a secure FSS scheme. If the hash function is efficient, such as an MD-variant (see, e.g., [DBP] and its references for possibilities and dangers), the resulting FSS scheme is practical. In particular, whereas the construction from bit commitments is only a one-time signature scheme, the construction from hash functions is not. This construction is also theoretically important because it is not known whether such hash functions are implied by or imply the existence of one-way permutations. Before, the weakest assumption known to imply FSS schemes was the existence of claw-free pairs of permutations.

Conversely, we show that any FSS scheme with a property we call the *almost unique secret key property* can be transformed into a statistically hiding bit commitment scheme. This property means that it is infeasible for a signer to compute more than one significantly different secret key corresponding to her public key; see below for details. All previously known FSS schemes have this property. Finally, we show that the existence of FSS schemes with this property is in fact *equivalent* to the existence of statistically hiding bit commitments with noninteractive opening and public verifiability.

1.3. *Organization of the Paper*

Section 2 describes the concepts that are used in this paper. Sections 3 and 4 contain the constructions of FSS and bit commitment schemes, respectively. Finally, Section 5 contains open problems.

2. Definitions and Notation

This section first introduces the model and recalls some information theory. Then the main concepts of this paper are described: fail-stop signatures, bit commitments plus universal and collision-intractable hash functions. Subsection 2.6 finally recalls some privacy amplification results.

2.1. *Model*

The basic model of computation used in this paper is based on Turing machines, mostly probabilistic. Thus saying that an algorithm can be executed in polynomial time means that there is a probabilistic Turing machine implementing this algorithm, which stops after polynomial time (in the length of the input or a security parameter given in unary).

Each participant of a protocol is modeled by an interactive Turing machine as defined in [GMR2]. This is a Turing machine equipped with a read-only input tape, a work tape, a random tape, and two communication tapes. One communication tape is read-only and used for receiving messages, and the other is write-only and used for sending messages.

An interactive protocol is a pair of interactive Turing machines sharing their communication tapes. The view of a participant, A, in an execution of an interactive protocol with B is defined as all the random bits used by A plus all the messages sent and received in this execution. The distribution of the corresponding random variable is induced by the random bits used by A and B. We refer to [GMR2] for a detailed definition of these concepts.

$\tilde{X}$ will denote any machine playing the role of X in a given protocol, but not necessarily following the prescribed methods.

2.2. *Information-Theoretic Preliminaries*

We will sometimes need notation and simple formulas from information theory. We briefly sketch them here. Interested readers are referred to [G] for more details and proofs.

Let X be a finite random variable taking values x with probabilities $Prob[x]$. The entropy of X is defined as

$$H(X) = -\sum_x Prob[x] \log_2(Prob[x]).$$

In particular, the entropy of a binary random variable that takes one of its two values with probability p is the following function of p:

$$H_{bin}(p) = -p \log_2(p) - (1-p) \log_2(1-p).$$

This function has its maximum at $H_{bin}(\frac{1}{2}) = 1$ and the gradient

$$H'_{bin}(p) = -\log_2(p) + \log_2(1-p),$$

which is monotonic decreasing.

The conditional entropy of X given another random variable Y is defined as the average of the entropies of the conditional distribution of X given a particular value

$Y = y$, written as

$$H(X|Y) = -\sum_{y} Prob[y]H(X|Y = y).$$

The mutual information between X and Y is defined as

$$I(X; Y) = H(X) - H(X|Y).$$

The mutual information is symmetric, i.e., $I(X; Y) = I(Y; X)$. Conditional mutual information $I(X; Y|Z)$ is derived in the same way from conditional entropies:

$$I(X; Y|Z) = H(X|Z) - H(X|Y, Z).$$

Entropies and information involving joint random variables, e.g., (X, Y), are related to simpler terms by

$$H(X, Y|Z) = H(X|Z) + H(Y|X, Z)$$

and

$$I(X, Y; W|Z) = I(X; W|Z) + I(Y; W|X, Z).$$

Finally, for any function f,

$$H(f(X)|X) = 0.$$

Combined with the previous rules, this gives many other rules about functions of random variables.

2.3. *Fail-Stop Signatures*

We first give a brief overview of the definition and then present the details; we also mention results about the relation to definitions of ordinary digital signature schemes. Finally, we define the almost unique secret key property.

2.3.1. *Overview of the Definition*

An FSS scheme consists of five components, all polynomial-time:

- a key generation protocol,
- an algorithm *sign* for signing,
- an algorithm *test* for verifying signatures; a signature passing this test is called *acceptable*,
- an algorithm *prove* for constructing proofs of forgery, and
- an algorithm *verify* for verifying proofs of forgery; a proof satisfying this predicate is called *valid*.

The fact that key generation is a protocol, and not carried out by the signer alone, is necessary to ensure that the signer does not generate a key pair for which she can prove her own signatures to be forgeries.

Obviously, an FSS scheme must satisfy that correct signatures are acceptable if the keys were generated correctly. Moreover, we require that correctly generated proofs of

forgery are valid. The more interesting parts of the definition are:

- *Security for the recipient*: It is infeasible for a polynomial-time bounded signer to produce an acceptable signature and a valid proof that it is forged.
- *Security for the signer*: It is impossible even for a computationally unbounded forger to produce a signature which the signer cannot prove to be a forgery (except with an extremely small probability).

2.3.2. *Details of the Definition*

For the details of the definition, we mostly follow [PP], with some details from [P2].

A fail-stop signature scheme has two security parameters: k for the security for the recipient and σ for the security for the signer. Furthermore, as in [GMR1], there is a parameter N, called the message bound, that limits the number of signatures that can be made with a certain key pair; if no bound needs to be known *a priori*, we use $N = \infty$ and define $1^{\infty} =$ '', the empty string.

Key generation is a two-party protocol executed by the signer A and a center B trusted by the recipients. Both parties have the three parameters (k, σ, N) as inputs. We remark that one can always do without a center, at some expense of efficiency depending on the individual scheme, basically by letting all the recipients share the role of B (see [PP, Section 3.3], and [P2]). As the output of key generation, A obtains a secret key sk, and a corresponding public key pk is output on a common broadcast channel. Instead, A or B may reject. If both parties are honest, this should only happen with negligible probability. If neither A nor B rejects, we say that the keys are accepted.

The algorithms *sign* **and** *prove*, which are carried out by the signer, may be probabilistic and use memory. The definition covers all cases by regarding all the random bits the signer uses as a part of the secret key (although algorithmically, they will often be generated much later), and signing as a deterministic function of the secret key and the sequence $\underline{m} = (m_1, \ldots, m_i)$ of the messages signed so far. Thus, in general, $sign(sk, i, \underline{m})$ denotes the signature on m_i if the previously signed messages were $m_1, \ldots, m_{i-1}$. The signatures in our constructions, like in most previous ones, only depend on the *number* of previously signed messages; in this case, we will write $sign(sk, i, m_i)$. The proof of a forgery is denoted by $prove(sk, m, s, hist)$, where m is a message, s a supposed signature on it, and *hist* the sequence of previously signed messages with their signatures. This result is either a bit string *proof* or the value "*not a forgery*."

The algorithms *test* **and** *verify* only depend on the public key and are deterministic and memoryless. Hence they can be carried out by anyone. We write $test(pk, m, s) = ok$ or *notok* and $verify(pk, m, s, proof) = accept$ or *reject*. We require that correctly generated proofs of forgery are valid, i.e., *verify* accepts any output different from "*not a forgery*" computed by an honest signer using *prove*.

Security for the recipient considers the following scenario: First, the cheating signer carries out key generation with B using an algorithm $\tilde{A}$. Then, using the view $view_{\tilde{A}}$ of $\tilde{A}$ from this protocol execution, the signer constructs a triple $(m, s, proof)$, using an algorithm $\tilde{A}^*$. (Note that the center would carry out key generation with different signers completely independently and sequentially, and that no active attacks on recipients are possible.)

Definition 2.1. A fail-stop signature scheme is secure for the recipient iff for all $\sigma, N, c > 0$ and all probabilistic polynomial-time algorithms, $\tilde{A}$, $\tilde{A}^*$, as described above, the following holds for k sufficiently large: The probability that the keys are accepted **and** s is an acceptable signature on m for the resulting pk **and** *proof* is a valid proof of forgery is at most k^{-c}.

This probability is over the random bits used by B, $\tilde{A}$, and $\tilde{A}^*$.

Security for the signer must hold even if the center cooperates with future recipients. Thus we consider a computationally unbounded cheating center, $\tilde{B}$. First A and $\tilde{B}$ execute the key generation protocol. $\tilde{B}$'s view of this protocol is denoted by $view_{\tilde{B}}$. Next, the signer signs some arbitrary messages; this results in a history *hist*. Finally, the enemy, given $view_{\tilde{B}}$ and *hist*, selects a pair (m, s) as a forgery.

Definition 2.2. Let a fail-stop signature scheme and a cheating center $\tilde{B}$ be given.

1. A value $hist = ((m_1, \ldots, m_i), (s_1, \ldots, s_i))$ is called a *possible history* for a given secret key sk if each s_j equals $sign(sk, j, (m_1, \ldots, m_j))$.
2. The set of possible secret keys (from the point of view of a computationally unbounded forger), given his view $view_{\tilde{B}}$ from key generation and a history *hist*, is denoted as $SK(view_{\tilde{B}}, hist)$. It consists of the secret keys sk' that are possible outcomes of key generation together with $view_{\tilde{B}}$, and where *hist* is a possible history for sk'. $SK(view_{\tilde{B}}, hist)$ is equipped with a probability distribution induced by the random bits used by A.
3. A *successful forgery* with respect to pk and $hist = ((m_1, \ldots, m_i), (s_1, \ldots, s_i))$ is a pair (m, s) where s is an acceptable signature on m with respect to pk and m does not occur in the history *hist*.
4. The set *Good* is defined as the set of outcomes of key generation where the signer will be able to prove any forgery after any history with very high probability. (This is a very strong definition—in particular, the history may have been generated by any kind of active attack.)
   More precisely, *Good* is the set of triples $(sk, pk, view_{\tilde{B}})$ such that for all possible histories *hist* (for the given sk) and all successful forgeries (m, s) (for the given pk and *hist*), with probability at least $1 - 2^{-\sigma}$ the signer obtains a valid proof of forgery by computing $prove(pk, m, s, hist)$. The probability is over the possible secret keys sk' in $SK(view_{\tilde{B}}, hist)$. (The "real" secret key, sk, only had to be mentioned to define the possible histories given this outcome of the key generation. From then on, everything is seen from the point of view of the forger.)
5. The scheme is *secure for the signer* if the probability that the keys are accepted and $(sk, pk, view_{\tilde{B}}) \notin Good$ is at most $2^{-\sigma}$. (The probability is over the random bits of A and $\tilde{B}$.)

Note that we allow an exponentially small error probability in two properties where [PP] does not: the properties that, if both A and B are honest, key generation does not fail and its outcome lies in the set *Good*. This is covered by the general definition of fail-stop signatures in [P1] and [P2]; the "standard fail-stop signature schemes" in [P2] allow such an error in the second of these properties only. The only theorem from [PP]

that we cited so far, about several recipients playing the role of the center, also holds for this generalization; this can be seen even more easily from [P2].

2.3.3. *Security Against Forgery*

For interested readers, we mention here how the security of FSS schemes relates to that of ordinary digital signature schemes, as defined in [GMR1].

First, Theorem 3.1 in [PP] says that any secure FSS scheme is secure against existential forgery after an adaptive chosen-message attack by polynomial-time forgers who did not participate in key generation, i.e., whose input is only the globally known parameters and the public key resulting from a correct execution of key generation. This also holds for our slightly more general definition: the proof in [PP] generalizes, but it can also be seen even more easily from the similar proof in [P2]. Thus, given that the computational assumption is not broken, no proofs of forgery will be needed against such forgers.

Secondly, if one worries that the center may be able to forge signatures (although such forgeries can be proved), one can exploit the methods to let several mutually distrusting recipients play the role of the center: We let the signer and the center play the former role of the center. Then not even the center will be able to forge unless it breaks the computational assumption.

Finally, every FSS scheme can be used to construct an equally efficient and secure ordinary digital signature scheme by omitting the center and the proofs of forgery, setting $\sigma = k$, and letting the signer carry out both roles in key generation (similar to Theorem 3.2 in [PP]).

2.3.4. *Almost Unique Secret Key Property*

Intuitively, the security definitions imply that a cheating signer cannot compute too many secret keys that are possible given the public key. Otherwise, she could prove her own signatures to be forgeries by using one secret key to sign and another key in the proof. All fail-stop signature schemes in the previous literature have an idealized version of this property: No matter how a polynomial-time bounded signer executes the key generation, she cannot compute two different secret keys that are both possible given the public key. We call this the *unique secret key property*. In the following, we use a relaxed version of it, the *almost unique secret key property*: Although the signer might be able to find more than one secret key fitting a public key, she cannot find *significantly different ones*. Keys are "not significantly different" if they lead to equal signatures. Moreover, we need a mapping κ on the secret keys with the intuitive meaning that $\kappa(sk)$ is the part of sk that makes a difference in the signatures.

Definition 2.3. A fail-stop signature scheme with security parameters (k, σ) has the almost unique secret key property if there are a polynomial-time computable predicate *Fits* and a polynomial-time computable mapping κ with the following properties:

- If the signer follows the key generation protocol, the resulting secret and public key, sk and pk, always fulfil $Fits(sk, pk) = 1$.
- For all $c > 0$, for all probabilistic polynomial-time algorithms $\tilde{A}$, $\tilde{A}^*$, and for k sufficiently large: The probability that the keys are accepted when $\tilde{A}$ execute the

key generation protocol with B and then $\tilde{A}^*$, given the view $view_{\tilde{A}}$, can compute sk_1, sk_2 such that $\kappa(sk_1) \neq \kappa(sk_2)$ and $Fits(sk_1, pk) = Fits(sk_2, pk) = 1$ is less than k^{-c}.

This probability is over the random bits used by B, $\tilde{A}$, and $\tilde{A}^*$.

- If sk_1 and sk_2 satisfy $Fits(sk_1, pk) = Fits(sk_2, pk) = 1$ and $\kappa(sk_1) = \kappa(sk_2)$, then for any message sequence, the signatures produced with sk_1 equal those produced with sk_2.

For a concrete FSS scheme, there will typically exist a function that computes the public key from a secret key. Then *Fits* can be constructed from this function. Furthermore, note that if κ is the identity, the third property is no restriction, and one obtains the unique secret key property.

Remark. All schemes previously proposed in the literature (see [PW1], [PW2], [HP], and [HPP]) have the almost unique secret key property, although one can easily construct artificial schemes without it. For instance, start with any secure FSS scheme, extend each secret key by a randomly chosen bit at the end, and append this bit to all signatures made with that key: In the sense of the above definition, two secret keys that differ only in the last bit are significantly different, but the signer can easily construct one from the other.

2.4. *Bit Commitments*

We define a bit commitment scheme as a pair of two-party protocols, namely the *commit* and the *opening* protocol. They take place between parties A and B, where A is the party committing herself.

- For the commit protocol, A gets as input a bit b, and both parties get a security parameter k. The concatenation of all the messages sent in the commit protocol is called *the commitment*. In some concrete schemes, it makes sense to define the commitment as a subset or a function of the messages; we have chosen our definition for simplicity. A or B may reject in the commit protocol, but if both parties are honest, this should only happen with negligible probability.
- For the opening protocol, A gets as input her view of the commit protocol, while B gets the commitment as input. At the end of the opening protocol, B outputs *reject*, *accept* 0, or *accept* 1. The intuitive meaning is that either B has detected cheating by A, or he accepts that A has opened the commitment to reveal either 0 or 1.

We will only consider commitment schemes with *noninteractive opening*, i.e., where the opening protocol consists of A sending one message to B, and where B's subsequent verification is deterministic.

Without loss of generality, we assume that no string can be accepted by B as both revealing 0 or 1, not even with exponentially small probability. For instance, this can be achieved if A sends the bit she wants to reveal as the first bit of her string.

We have also built into the model a second property, which we call *public verifiability*: B can verify the opening based on the commitment only. This means that anyone who trusts that a given commitment is the result of a conversation with B can verify the opening

without knowing B's random bits. This property is necessary in the construction of an FSS scheme to ensure that everybody can verify signatures.

Obviously, a BC scheme must satisfy that if both parties are honest and do not reject the commit protocol, B will accept the bit that A committed to. The more interesting parts of the definition are the binding and the security property.

Definition 2.4. A pair of protocols as described above is called a bit commitment scheme with noninteractive opening (BC scheme) if it has the following *binding property*.

Let $\tilde{A}$ be any polynomial-time bounded machine that executes the commit protocol with B and then outputs two strings s_0 and s_1. (With these strings, the cheating committer hopes to have the choice between opening the commitment to reveal 0 by using s_0, or to reveal 1 by using s_1.) Let $p(\tilde{A}, k)$ be the success probability of $\tilde{A}$, i.e., the probability that B did not reject the commit protocol, *and* that for both $b = 0$ and $b = 1$, B outputs *accept* b on input s_b in the opening protocol. The probability is taken over the random bits of $\tilde{A}$ and B. Then $p(\tilde{A}, k) < k^{-c}$ for all $c > 0$ and k sufficiently large.

We now give two definitions of the property that a BC scheme is statistically hiding. We show below that they are equivalent except for parameter transformations. The different formulations will be useful in different proofs. The two definitions, and their equivalence, can be extended to multibit commitments, see [DPP].

Definition 2.5. A BC scheme is called statistically hiding if it has the following *bias-based security property*: Let any $\tilde{B}$ be given, and let b_δ denote the random variable of the bit committed to if $b = 0$ with probability δ. Let $bias_\delta(v)$ denote $\tilde{B}$'s advantage in guessing b given $\tilde{B}$'s view v of the commit protocol, i.e.,

$$bias_\delta(v) = |\delta - Prob[b_\delta = 0 \mid v]|.$$

Then the expected value $Bias_\delta$ of $bias_\delta(v)$ is at most 2^{-k} for all δ. The probabilities are taken over the choice of b_δ and the random bits of A and $\tilde{B}$.

Note that the definition means that δ is known to $\tilde{B}$; in other words, $\tilde{B}$ can have arbitrary *a priori* information about the bit committed to.

Definition 2.6. A BC scheme is said to have the *capacity-based security property* if the following holds: Let any $\tilde{B}$ be given. Then the commit protocol defines transition probabilities from the bits b to the views v of $\tilde{B}$. If it is considered as a channel with b as an input and v as an output, its channel capacity $C_{\tilde{B}}$ is

$$C_{\tilde{B}} = \max_{\delta}(I(V_\delta; b_\delta)),$$

where V_δ denotes the random variable of $\tilde{B}$'s view in the case where $b = 0$ with probability δ. Then $C_{\tilde{B}}$ is at most 2^{-k}.

This is a natural information-theoretic definition of "hiding," because it means that for any *a priori* information, $\tilde{B}$'s view from the commit protocol only gives negligible additional information about b.

Lemma 2.7. *Let any BC scheme and any $\tilde{B}$ be given. Then for all δ:*

$$Bias_\delta \leq Bias_{1/2}.$$

Thus $\delta = \frac{1}{2}$ is the worst case on average.

The proof of this lemma is in the Appendix. Note that for individual views the best we can show is $bias_\delta(v) \leq 2bias_{1/2}(v)$.

Lemma 2.8. *The bias-based security property and the capacity-based security property of BC schemes are equivalent, except for small transformations of the security parameters. More precisely:*

1. *If a BC scheme has the bias-based security property, it has a channel capacity of at most $k2^{-k/2}$ for any $\tilde{B}$ and $k \geq 4$.*
   Thus, a capacity of at most 2^{-k} can be achieved by using a security parameter k' with $k' \geq 4$ and $k'/2 - \log_2 k' \geq k$.
2. *If a BC scheme has the capacity-based security property, it has an average bias of at most $2^{-k/2}$ for any $\tilde{B}$.*
   Thus, an average bias of at most 2^{-k} can be achieved by using the security parameter $k' = 2k$.

The proof of this lemma is in the Appendix.

Corollary 2.9. *In order to prove the capacity-based security property of a BC scheme, it is sufficient to show that $I(V_{1/2}; b_{1/2}) \leq 2^{-k'}$, where k' is such that $k' \geq 8$ and $k'/4 + 1 - \log_2 k' \geq k$ (i.e., to consider the case $\delta = \frac{1}{2}$).*

The proof is in the Appendix.

Remark. We have required an exponential decrease of the bias in the bias-based security property. This makes our construction of bit commitments from FSS schemes stronger. For the converse construction, it is not a significant restriction, because standard "XOR-ing" techniques can be used to improve weaker schemes so that they satisfy the definition: In order to commit to one bit b, choose a random string of bits (of length $n = c \cdot k$ for some constant c) with $b = b_1 \oplus \cdots \oplus b_n$, and commit to each bit b_i with the weaker scheme. $\tilde{B}$'s advantage in guessing b decreases exponentially in k (see, e.g., Scheme (ii), Section I, of [W]). Moreover, most practical examples known have a bias of 0, and in particular the construction from one-way permutations that we will use below.

2.5. *Hash Functions*

Two very different types of hash functions will be used in the following: Collision-intractable (collision-free) hash functions and universal hash functions.

Collision-intractable hash functions were formally defined in [D1]. This definition is sketched first.

Definition 2.10. Let $o(m)$ be a function $\mathbb{N} \to \mathbb{N}$ such that $o(m) < m$. Let H be a family of finite sets $\{H_m\}_{m \in \mathbb{N}}$ such that each member of H_m is a function $\{0,1\}^m \to \{0,1\}^{o(m)}$. H is called a family of collision-intractable hash functions if the following holds:

1. Given m, a function in H_m can be chosen at random in polynomial time in m.
2. Given $h \in H_m$ and $x \in \{0,1\}^m$, the value $h(x)$ can be computed in polynomial time in m.
3. For all $c > 0$: If $h \in H_m$ is selected as in 1, no probabilistic polynomial-time algorithm can find $x, y \in \{0,1\}^m$ such that $x \neq y$ and $h(x) = h(y)$ with probability greater than m^{-c} for m sufficiently large. The probability is over the selection of h and the random choices of the collision-searching algorithm.

We can build a similar family with any desired input length from any collision-intractable family by fixing some input bits if the input length is too large, and using the iterative construction of [D2] if inputs are too short.

Universal hash functions were defined in [CW].

Definition 2.11. A class F of functions $A \to B$, where A and B are finite sets, is called universal$_2$ (or simply universal) if for any distinct $a_1, a_2 \in A$ the probability that $f(a_1) = f(a_2)$ is at most $1/|B|$, when f is chosen uniformly at random in F.

In practice we shall need not only a single class of functions, but a family $F = \{F_n\}_n$ of classes of functions $\{0,1\}^n \to B_n$ such that:

1. Given n, a function in F_n can be selected uniformly at random in polynomial time in n.
2. Given $f \in F_n$ and $a \in \{0,1\}^n$, the value $f(a)$ can be computed in polynomial time in n.
3. Every class F_n is universal.

In order to follow previous definitions, we have required that universal hash functions be chosen uniformly at random. However, the definition can be extended to other distributions by requiring the same distribution in Parts 1 and 3 of the definition.

These functions are interesting because they emulate some properties of random functions, although they have much shorter descriptions, and can therefore be efficiently used in protocols. The standard example of a family of universal hash functions from n-bit strings to i-bit strings with $i \leq n$ are the functions given by

$$x \mapsto (ax + b)|_i,$$

where $|_i$ means that we take only the most significant i bits, and where $a, b \in GF(2^n)$. Thus each member of the family is characterized by a choice of a and b, which requires $2n$ bits.

2.6. *Privacy Amplification with Universal Hash Functions*

In [BBR] and [BBCM] it is shown how universal hash functions can be used to hide information about a string. These papers use the notion of collision (or Renyi) information. We note that collision information has some counterintuitive properties, but this

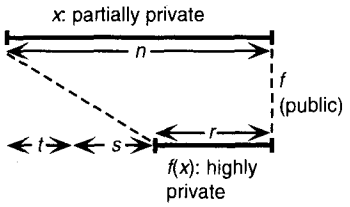

Fig. 1. Privacy amplification. The enemy has $t = n - R(X)$ bits of collision information about x. $s = R(X) - r$ is a security margin.

is no problem in our case because we only use it in technical lemmas, and not in our security definitions.

Let X be a finite random variable taking the value x_i with probability $p_i \geq 0$ for $i = 1, 2, \ldots, 2^n$. The collision entropy of X is defined as

$$R(X) = -\log_2 \left(\sum_1^{2^n} p_i^2 \right)$$

and the collision information of X as

$$I(X) = n - R(X).$$

Using Jensen's inequality (see, e.g., [F]) it can be shown that $H(X) \geq R(X)$. We need the following privacy amplification theorem from [BBCM]; it shows how, given a string about which an enemy has partial information, one can extract a shorter string that is almost completely unknown to the enemy (see Fig. 1).

Theorem 2.12. *Let a random variable, X, with Renyi entropy $R(X)$, on the set of n-bit strings be given. Let $0 < r \leq n$ and let F be a universal class of hash functions from $\{0, 1\}^n$ to $\{0, 1\}^r$. If f is chosen uniformly at random in F, the expected amount of collision information (over the choice of f) about $f(X)$ given f is at most $\log_2(1 + 2^{r-R(X)}) \leq 2^{r-R(X)}/\ln 2$ bits.*

This is a restatement of [BBCM, Theorem 3] restricted to Renyi information and with a notation fitting our needs. We derive the following lemma, which is an application of privacy amplification to a random variable where an upper bound on the probability of each single outcome is known.

Lemma 2.13. *Let a random variable, X, on the set of n-bit strings and $\alpha > 0$ be given, such that $p_i \leq \alpha$ for the probability p_i of each string x_i. Let F be a universal class of hash functions from $\{0, 1\}^n$ to 1 bit. If f is chosen uniformly at random in F, the expected amount of collision information (over the choice of f) about $f(X)$ given f is at most $\alpha(2/\ln 2)$.*

Proof. Let $N = 2^n$. Then

$$\sum_{i=1}^{N} p_i^2 \leq \sum_{i=1}^{N} p_i \alpha = \alpha.$$

Hence the collision entropy $R(X)$ of the given distribution is

$$R(X) \geq -\log_2(\alpha).$$

Using Theorem 2.12 with $r = 1$ implies that the enemy's expected collision information E about $f(x)$ is

$$E \leq \frac{2^{1+\log_2(\alpha)}}{\ln 2} = \alpha \cdot \frac{2}{\ln(2)}. \qquad \square$$

3. Constructions of Fail-Stop Signatures

In this section, we present constructions of FSS schemes. The first one is based on a statistically hiding BC scheme as defined above. The basic idea is very similar to Lamport and Diffie's one-time signatures (see [L] and [DH]). Next, this construction is extended so that an FSS scheme with the almost unique secret key property is obtained. Finally, an FSS scheme is constructed given collision-intractable hash functions. This scheme has several efficient variants to sign many long messages.

3.1. *Fail-Stop Signatures from Bit Commitments*

Informally, our idea is as follows: the signer A makes a one-time key by making two commitments C_0, C_1 to bits b_0, b_1. To sign a message bit m, he opens C_m. By the binding property, he can sign a bit in only one way, and by statistical hiding, no adversary can guess b_0, b_1. With probability 50% , a forgery will therefore provide A with another way of opening C_0 or C_1. This probability can be boosted to almost 1 by having many pairs of commitments.

The formal description: Recall that an FSS scheme has two security parameters, k and σ, and a message bound N. We will assume for simplicity that an upper bound, L, is known on the length of each message to be signed. In this case we can, without loss of generality, assume that every message has length precisely L (shorter messages can be padded to length L using a method that allows reconstruction of the original message). For our construction, we will let $U = NL$, such that U is the total number of bits to be signed.

We remark that it is easy to modify our scheme to deal with the case where no upper bound on the length of the individual messages is known *a priori*. As our scheme works by signing messages bit by bit, this only requires that messages are coded such that the start and end of a message can be identified.

In contrast to the construction from hash functions in Section 3.3 and all the previous constructions in the literature, we have no way of shortening long messages before signing. No such method is known under the sole assumption of the existence of a BC scheme. Similarly, this scheme is a pure one-time scheme, whereas in all the other constructions, e.g., the length of the public key is independent of N. In particular if the given BC scheme needs interaction for every new commitment, as that in [NOVY] does, it seems hard to find a way around this.

Protocol 1

KEY GENERATION

The center B and the signer A execute $4U\sigma$ instances of the commit protocol with security parameter $k' = \max(k, \sigma + \log_2(\sigma) + \log_2(U) + 5)$. For each instance, A chooses randomly and uniformly the bit to commit to. The resulting commitments are organized in U pairs of lists $(C_{n,0,1}, \ldots, C_{n,0,2\sigma})$ and $(C_{n,1,1}, \ldots, C_{n,1,2\sigma})$ for $1 \leq n \leq U$. The public key pk consists of all these commitments, while the secret key sk consists of the $4U\sigma$ strings known to A that will open the commitments. These strings are denoted by $s_{n,0,i}$ and $s_{n,1,i}$, respectively. Either party stops and rejects the keys if they reject during any of the commit protocols.

SIGNING

The signature on the jth message, $m = b_{L(j-1)+1} \cdots b_{Lj}$, is defined as

$$Sign(sk, j, m) := (sign(sk, n, b_n))_{n=L(j-1)+1,\ldots,Lj} \qquad (1 \leq j \leq N),$$

where the 1-bit signature $sign(sk, n, b)$ is defined as

$$sign(sk, n, b) := (n, s_{n,b,1}, \ldots, s_{n,b,2\sigma}),$$

i.e., A opens one of the two lists of commitments prepared for the nth bit.

VERIFICATION

To verify a signature $S = (S_i)_{i=1,\ldots,L}$ on $m = b_1 \cdots b_L$, one verifies that the sequence of bit numbers in the S_i's is of the form $(L(j-1)+1, \ldots, Lj)$ for some j and each S_i is the correct 1-bit signature on b_i. To verify a 1-bit signature, $(n, s_{n,b,1}, \ldots, s_{n,b,2\sigma})$, on a bit b, one verifies that the 2σ strings in the signature open the commitments $C_{n,b,1}, \ldots, C_{n,b,2\sigma}$ from the public key correctly. (Note that the bits revealed in this process are irrelevant; in particular, they have no relation with b.)

PROOF OF FORGERY

Given an acceptable signature on a message, m, the signer tries to generate a proof that one of the 1-bit signatures is a forgery and outputs the first proof that is generated. Such a proof is generated as follows: Given an acceptable signature s on a bit b, A generates her own signature $s' = sign(sk, n, b)$ on b, where n is the same bit number as in s. She searches for an i $(1 \leq i \leq 2\sigma)$ for which the ith bit revealed in s is different from the ith bit revealed in s'. If such an i is found, she outputs n, b, i and the two strings used to open this commitment, i.e.,

$$proof := (n, b, i, s_{n,b,i}, s'_{n,b,i}).$$

If not, she outputs "*not a forgery*."

VALIDATING PROOF OF FORGERY

A tuple, $(n, b, i, s_{n,b,i}, s'_{n,b,i})$ is a valid proof of forgery for a signature S on a message m if:

- S is acceptable;
- n is the index of some bit, b, signed in S;

- $1 \leq i \leq 2\sigma$;
- $s_{n,b,i}$ is the ith string in the 1-bit signature on b; and
- $s_{n,b,i}$ and $s'_{n,b,i}$ open $C_{n,b,i}$ to reveal different bits.

Theorem 3.1. *If the underlying statistically hiding BC scheme (with public verifiability and noninteractive opening) is secure, Protocol 1 is a secure fail-stop signature scheme.*

Proof. Correct signatures are obviously acceptable, and correctly generated proofs of forgery are valid.

Security for the recipient. Assume, in contradiction to Definition 2.2, that there are values σ, $N > 0$ and probabilistic polynomial-time algorithms $\tilde{A}$, $\tilde{A}^*$ where $\tilde{A}$ carries out the key generation protocol with the correct center B and $\tilde{A}^*$ then produces values m, s, and *proof* such that, with nonnegligible probability, the keys are accepted, s is an acceptable signature on m, and *proof* is a valid proof of forgery.

By the definition of key generation and the validation of proofs of forgery, this yields an algorithm $\tilde{A}'$ that succeeds, with nonnegligible probability, in carrying out $4U\sigma$ instances of the commit protocol and then opening at least one of the commitments, say $C_{n,b,i}$, to reveal two different bits.

This rather obviously contradicts the binding property of the commitment scheme. However, for completeness we show that the choice among $4U\sigma$ commitments does not make the cheater's task too easy. We construct an algorithm $\tilde{A}''$ contradicting Definition 2.4 as follows: On input k, where $k \geq \sigma + \log_2(\sigma) + \log_2(U) + 5$ without loss of generality, $\tilde{A}''$ randomly chooses a triple (n, b, i) $(1 \leq n \leq U, b \in \{0, 1\}, 1 \leq i \leq 2\sigma)$. It then calls $\tilde{A}'$ with parameters (k, σ, N). In the $4U\sigma$ executions of the commitment protocol that $\tilde{A}'$ now wants to carry out, $\tilde{A}''$ itself plays the role of B, to whom the commitment is made, except for the execution with index (n, b, i), which is carried out with the real B. Now, if $\tilde{A}'$ succeeds in computing a valid proof of forgery, $\tilde{A}''$ finds at least one index (n', b', i') such that $\tilde{A}'$ has output two strings $s_{n',b',i'}$ and $s'_{n',b',i'}$ that open the commitment $C_{n',b',i'}$ in two ways. If $(n', b', i') = (n, b, i)$, the algorithm $\tilde{A}''$ outputs $(s_{n',b',i'}, s'_{n',b',i'})$. This happens with probability $1/4U\sigma$ because of the random choice of (n, b, i). Thus the success probability of $\tilde{A}''$ is $1/4U\sigma$ times that of $\tilde{A}'$. This is still nonnegligible, because $U = NL$ and σ are constant with respect to k (recall the order of the quantifiers in Definition 2.1).

Security for the signer. Let a cheating center $\tilde{B}$ be given. Let Acc denote the event that the keys are accepted, and let G denote the good event that the bias of all the bits committed to, given the view of $\tilde{B}$, is less than $1/8$.

First, we show that the probability that A accepts the keys and G does not occur is very small: For a single commitment, by the bias-based security property and Markov's rule, the probability that the bias is larger than $1/8$ is at most $8 \cdot 2^{-k'} = 2^{-k'+3}$. Moreover, the $4U\sigma$ executions of the commitment protocol are carried out with independent random choices by A (including the bits committed to). We therefore obtain that

$$Prob[Acc, \neg G] \leq 1 - (1 - 2^{-k'+3})^{4U\sigma} < 4U\sigma 2^{-k'+3} \leq 2^{-\sigma}.$$

Second, we show that G implies that $(sk, pk, view_{\tilde{B}}) \in Good$. Let a possible history *hist* for sk and a successful forgery (m, S) be given. By the definition of a successful forgery, m was not signed by the real signer. Thus, although the enemy has seen many of the commitments opened, there is at least one index n and a bit b in m for which S contains a 1-bit signature on b with index n and *hist* does not contain any information about the contents of the commitments $(C_{n,b,1}, \ldots, C_{n,b,2\sigma})$ (by the independent random choices of the signer). Hence, by the definition of G, the enemy still cannot guess any of their contents with probability better than 5/8, and if he has to guess all these contents, he will succeed with probability at most $(5/8)^{2\sigma} \leq 2^{-\sigma}$. To predict A's correct signature on b as the nth bit, this is exactly what the enemy has to do: If he guesses any of these contents wrong, the algorithm *prove* succeeds. Thus A can in fact prove any forgery with probability at least $1 - 2^{-\sigma}$.

The probability that the keys are accepted and $(sk, pk, view_{\tilde{B}}) \notin Good$ is therefore at most $2^{-\sigma}$, i.e., Definition 2.2 is fulfilled. □

Corollary 3.2. *If one-way permutations exist, then there exists a secure FSS scheme.*

Proof. In Section 3.1 of [NOVY], a statistically (in fact, perfectly) hiding bit commitment scheme is constructed from any one-way permutation. One sees immediately from the construction that it has public verifiability and noninteractive opening. Hence it can be used in Theorem 3.1. □

3.2. *Schemes with the Almost Unique Secret Key Property*

The FSS scheme constructed from bit commitments in Section 3.1 does not necessarily have the unique secret key property. For example, more than one bit string may be acceptable as opening a commitment as a 1, and the definition of bit commitments does not exclude that a committer knows more than one such string. In the following, we modify the scheme so that it has at least the almost unique secret key property. Together with Theorem 4.1, this will yield an equivalence between FSS schemes with the almost unique secret key property and statistically hiding BC schemes with noninteractive opening and public verifiability.

As a first step, we use any commitment scheme satisfying our definition to build a new commitment scheme where the committer is committed to both a bit and, to some extent, to the way in which he will open the commitment. The idea is very simple: First commit to a bit b_0 using a commitment C_0, and then make additional commitments to the bits in the string you will use to open C_0. What has to be proved is that the additional commitments do not help the receiver to guess b_0.

Thus, in the following protocol, we assume that we are given a secure BC scheme with public verifiability and noninteractive opening. Without loss of generality, we may assume that a commitment made in this scheme with security parameter value k can be opened with a string of length precisely $p(k)$, where p is a polynomial. We now describe the new commitment scheme with security parameter k.

Protocol 2

COMMIT

Let b_0 be the bit A wants to commit to. A first makes a *primary* commitment C_0 to b_0. This is done with security parameter k' chosen as the smallest integer for which $k' - \log_2(1 + p(k')) \geq k$. Let $s_0 = b_1, ..., b_{p(k')}$ be the string that A can use to open C_0 correctly. Now A makes $p(k')$ *secondary* commitments $C_1, \ldots, C_{p(k')}$ with security parameter k', where C_i is a commitment to b_i.

OPENING

A opens the commitment by sending s_0 to B, who verifies that s_0 opens C_0 correctly. If this is the case, he outputs *accept* b_0, otherwise *reject*.

It may seem strange that the secondary commitments are never opened in the protocol above, but they are only present for the purpose of the construction of an FSS scheme to follow.

Lemma 3.3. *The commitment scheme described in Protocol 2 is a BC scheme with public verifiability and noninteractive opening, and it has the capacity-based security property if the underlying scheme has it.*

By Lemma 2.8, we obtain a similar result for bias-based security by first increasing the security parameter of the underlying scheme, and then that of the new scheme bias-based.

Proof. Public verifiability and noninteractive opening follow trivially from the corresponding properties of the underlying commitment scheme. This is also the case for the binding property, because the construction forces A to make the primary commitment with a security parameter $k' > k$.

For the capacity-based security property, we estimate the amount of information that $\tilde{B}$'s views of the $p(k')+1$ commit protocols give about b_0. Let V_i be the random variable representing $\tilde{B}$'s view of the ith commit protocol. By the rules mentioned in Section 2.2, and because we required b_0 to be a function of s_0:

$$\begin{aligned} I(V_0 \ldots V_{p(k')}; b_0) &= I(V_0; b_0) + I(V_1; b_0|V_0) + \cdots + I(V_{p(k')}; b_0|V_0, \ldots, V_{p(k')-1}) \\ &\leq I(V_0; b_0) + I(V_1; s_0|V_0) + \cdots + I(V_{p(k')}; s_0|V_0, \ldots, V_{p(k')-1}). \end{aligned}$$

We next show that each term $I(V_i; s_0|V_0, \ldots, V_{i-1})$ equals $I(V_i; b_i|V_0, \ldots, V_{i-1})$, i.e., that the only new information one gets in the ith commitment protocol is about the ith bit. This is intuitively clear, because A executes this protocol with random choices, say r_i, that are independent of everything else so far.

More precisely, we assume without loss of generality that all the random bits used by $\tilde{B}$ are included in his original view V_0. We fix i and let $V' = V_0, \ldots, V_{i-1}$. Hence V_i is a function of r_i, b_i, V'. We have

$$\begin{aligned} I(V_i; s_0|V') - I(V_i; b_i|V') &= -H(V_i|s_0, V') + H(V_i|b_i, V') \\ &= I(V_i; s_0|b_i, V') \qquad \text{(note that } b_i \text{ is a part of } s_0) \\ &\leq I(V_i, r_i, b_i, V'; s_0|b_i, V') \end{aligned}$$

$$
\begin{aligned}
&= I(r_i, b_i, V'; s_0 | b_i, V') \\
&\qquad\qquad \text{(because } V_i \text{ is a function of } r_i, b_i, V') \\
&= I(r_i; s_0 | b_i, V') \\
&= H(r_i | b_i, V') - H(r_i | s_0, b_i, V') \\
&= H(r_i) - H(r_i) \\
&\qquad\qquad \text{(by the independent choice of } r_i) \\
&= 0.
\end{aligned}
$$

The third and fifth line follow easily from the formulas in Section 2.2. Substituting the result into the first inequality gives

$$
\begin{aligned}
I(V_0 \ldots V_{p(k')}; b_0) &\leq I(V_0; b_0) + I(V_1; b_1 | V_0) + \cdots + I(V_{p(k')}; b_{p(k')} | V_0, \ldots, V_{p(k')-1}) \\
&\leq (1 + p(k'))2^{-k'} \\
&\leq 2^{-k}.
\end{aligned}
$$

This establishes the required security property. □

Theorem 3.4. *If there exists a statistically hiding bit commitment scheme with public verifiability and noninteractive opening, there exists a secure fail-stop signature scheme with the almost unique secret key property.*

Proof. Consider the FSS scheme resulting from using the BC scheme in Protocol 2 as the basis of Protocol 1. This is secure by Lemma 3.3 and Theorem 3.1. To obtain the almost unique secret key property, we expand the secret key sk so that it also contains the strings that open all the secondary commitments made in the key generation. Nothing else in the scheme is changed, so that the extra information is never used in signing, verifying, or proofs of forgery. Hence the expansion does not affect the security of the FSS scheme.

We define $Fits(sk, pk)$ to be true if and only if the strings in the secret key sk open all the commitments in the public key pk correctly—primary as well as secondary ones. The significant part, $\kappa(sk)$, of sk, is defined to be the list of strings that open the primary commitments. Clearly, finding two secret keys fitting the same public key, but with different κ-images, would mean opening at least one secondary commitment to reveal two different bits. Details of the proof that opening one out of (polynomially) many commitments in two ways is infeasible with a secure commitment scheme can be seen in the proof of Theorem 3.1. Moreover, two secret keys sk_1, sk_2 with $\kappa(sk_1) = \kappa(sk_2)$ obviously lead to the same signatures—recall that the signature only contains strings that open primary commitments. □

3.3. *Fail-Stop Signatures from Hash Functions*

Now we construct far more efficient FSS schemes under the assumption that collision-intractable hash functions exist.

Assume we have a family, H, of collision-intractable hash functions mapping $(k + 2\sigma + 1)$-bit inputs to k-bit outputs. We make the following observation:

Lemma 3.5. *Let h be any function from $k + 2\sigma + 1$ bits to k bits. If x is uniformly chosen, the probability that the preimage set of $h(x)$ has at least 2^σ elements is at least $1 - 2^{-\sigma-1}$.*

Proof. Let the degree of a point in the image of h be the size of its preimage set under h. As h maps into the set of k-bit strings, at most $2^k \cdot 2^\sigma$ elements can be preimages of points of degree $\leq 2^\sigma$. Hence a uniformly chosen x is such a preimage with probability at most $2^{k+\sigma}/2^{k+2\sigma+1}$. □

We use this result to build a simple FSS scheme. Informally, we observe that a collision-intractable hash function h is also one-way and can therefore be used as the basis of a classical one-time signature scheme where one signs a bit by revealing one out of two possible preimages under h. But this scheme is also an FSS scheme since, first, collision intractability means that the signer can sign each bit in only one way and, second, the above lemma implies that guessing x given $h(x)$ is hard even for an unbounded adversary if the length of x is chosen properly.

As we have collision-intractable hash functions available, several known constructions for signing many long messages, given a one-time signature scheme, can be applied. The only difference is again that with a small probability, a key may not be good even if no party cheats in key generation. For completeness, we present and prove the simplest of these constructions entirely. This is bottom-up tree authentication, following [M1] and [M2], and for fail-stop signatures [PW2]; see Fig. 2. Other constructions are mentioned at the end of this section.

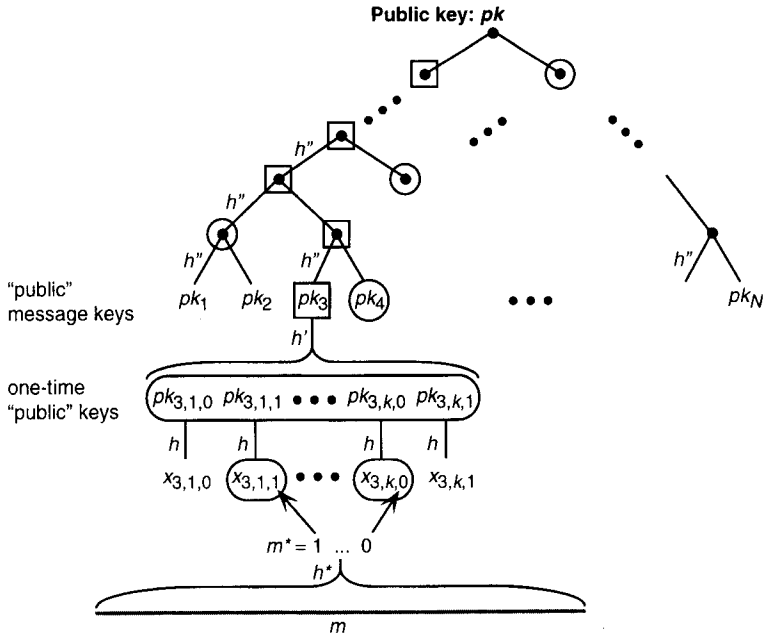

Fig. 2. Bottom-up tree authentication. The nodes in circles are the signature on the third message; the values of the nodes in squares are computed during verification.

Protocol 3

KEY GENERATION

- From the given parameters (k, σ, N), both parties compute $\sigma' := \sigma + \log_2(N) + \log_2(k)$.
- B chooses a hash function h from the given family H for the parameters (k, σ') and sends it to A. A verifies that h is a function from $k + 2\sigma' + 1$ bits to k bits.
- A chooses Nk pairs of preimages $(x_{n,l,0}, x_{n,l,1})$ as her secret key ($1 \leq n \leq N$, $1 \leq l \leq k$). (Each such pair will later serve to sign one bit.) She computes the values $pk_{n,l,b} := h(x_{n,l,b})$, which will be called one-time "public" keys. (They would be the public key if we only presented the one-time signature scheme; now, however, they are not published for reasons of efficiency.)
- As a special bottom layer of a hash tree, A hashes the one-time "public" keys that belong to the same message number as $pk_n := h'(pk_{n,1,0}, pk_{n,1,1}, \ldots, pk_{n,k,0}, pk_{n,k,1})$. Here, h' is a hash function from $2k^2$ bits to k bits. (Recall that such functions can be constructed iteratively from the given h, see Section 2.5 and [D2].) We call each pk_n a "public" message key.
- A constructs a binary hash tree over the "public" message keys, i.e., these keys are the values of the leaves of the tree. Then, bottom-up, the value of each node is computed by hashing the concatenation of the values of its two children. This time we need a hash function, h'', from $2k$ to k bits.
- A publishes the value of the root as her public key pk.

SIGNING

We define $sign(sk, n, m)$, where m is the nth message to be signed. First, m is hashed down to k bits with a function h^*, once again using the iterative construction from [D2]. The result and its bits are denoted by $m^* = (b_1, \ldots, b_k)$.

Next, a one-time signature is constructed as $s^* := (x_{n,1,b_1}, \ldots, x_{n,k,b_k})$, i.e., the preimages corresponding to the individual bits of m^*.

Finally, the one-time signature is authenticated with respect to pk: A collects all the one-time "public" keys corresponding to n, and all the immediate neighbours of the nodes in the path from pk_n to the root. The tuple of n, s^*, and these values is called the complete signature, s.

VERIFICATION

The recipient first also hashes m to obtain m^*. Moreover, he verifies the authentication path for the one-time "public" keys by recomputing all the hash values on the path to the root. The hash value in the root must equal pk. Finally, he verifies the one-time signature $s^* = (x_1, \ldots, x_k)$ with respect to the one-time "public" keys: For each bit of m^*, he verifies that $h(x_l) = pk_{n,l,b_l}$. (Recall that n is given as a part of the signature.)

PROOF OF FORGERY

Given a successful forgery (m, s), where the message number given in s is n, the signer first retrieves her own nth message m', if she has already signed n messages. If $m'^* := h^*(m') = h^*(m) = m^*$, she has found a collision of h^*, because $m \neq m'$, and outputs it as her proof of forgery.

Otherwise, she compares the one-time "public" keys in both signatures. If they are different, she follows the authentication paths from these keys to pk, and she finds a collision of h' or h'' with probability 1.

In the remaining case, the forged signature contains an acceptable one-time signature $s^* = (x_1, \ldots, x_k)$ with respect to the correct one-time "public" keys pk_{n,l,b_l}. If any of these values x_l is different from her own x_{n,l,b_l}, she outputs the pair of them (a collision of h, because both have the image pk_{n,l,b_l}), as a proof of forgery. Otherwise, she outputs "*not a forgery*."

VALIDATING PROOF OF FORGERY
Any collision of the given hash function h or the hash functions with different preimage lengths constructed from h counts as a proof of forgery.

Theorem 3.6. *If the underlying family H of hash functions is collision-intractable, Protocol 3 is a secure fail-stop signature scheme. The length of the public key is the security parameter k, independent of the message bound, N.*

Proof. Correct signatures are obviously acceptable, and correctly generated proofs of forgery are valid.

The *security for the recipient* follows immediately from the collision-intractability of H, because any proof of forgery includes a collision. (In case of doubt that this is true although several hash functions are derived from H, note that, by the proofs of the collision-intractability of those constructions, every collision of them can easily be transformed into a collision of the original H.)

Security for the Signer. Let the good event G be that all the one-time "public" keys have preimage sets of size at least $2^{\sigma'}$. By Lemma 3.5, and as all the preimages $x_{n,l,b}$ are chosen independently, the probability that the keys are accepted, but G does not occur, is

$$Prob[Acc, \neg G] \leq 1 - (1 - 2^{-\sigma'-1})^{2Nk} < 2Nk2^{-\sigma'-1} \leq 2^{-\sigma}.$$

We now show that G implies that the outcome of the key generation lies in *Good*. Let any possible history and any successful forgery (m, s) be given. If the hash value m^* of m equals m'^*, the hash value of the message the signer signed with the same message number n, or if the one-time "public" keys in s are not correct, the construction of a proof of forgery finds a collision with probability 1. In the remaining case, s contains an acceptable one-time signature $s^* = (x_1, \ldots, x_k)$ with respect to the correct one-time "public" keys for the message number n, and there is at least one index l $(1 \leq l \leq k)$ such that the lth bits of m^* and m'^*, say b_l and b'_l, are different. Thus the signer has not yet shown her preimage x_{n,l,b_l} of pk_{n,l,b_l}, and the forgery is provable unless the enemy has guessed correctly which of the at least $2^{\sigma'}$ preimages of pk_{n,l,b_l} the signer has. Moreover, the signer has not given any information about x_{n,l,b_l} apart from pk_{n,l,b_l}. (The remaining keys are statistically independent, and the hash tree and the history are deterministic functions of those values.) Hence the signer can in fact prove this forgery with probability at least $1 - 2^{-\sigma'} > 1 - 2^{-\sigma}$.

Length of the Public Key. The length of pk is obviously k. Only note that the function h can be the same for all signers, and thus it makes sense not to count its description as a part of the public key. (Such values are usually called prekeys, e.g., in [PP].) □

As extremely efficient hash functions exist in practical applications (e.g., MD-variants and constructions from block-ciphers, but see [DBP] again), this shows that really practical FSS schemes can be constructed based on conventional cryptography only.

Instead of bottom-up tree authentication, as just described, we can also use top-down tree authentication, see [PP] (or [HPP]). This means that each node signature is used to sign two new "public" message keys, similar, e.g., to [GMR1]. More precisely, Construction 5.1 in [PP] shows that any one-time FSS scheme with prekey where an arbitrarily long message can be signed yields an FSS scheme for any number of messages. In proving the same construction secure in our case, we only have to increase σ logarithmically to compensate the probability that any of the about $2N$ one-time key pairs is not good, just as in the bottom-up construction above. The public key is only as long as in the one-time scheme, and the signature length grows logarithmically in N, similar to the bottom-up construction. Variants of the construction are also sketched where no message bound needs not be known in advance. One can also use bottom-up trees within a top-down construction to combine the shorter signatures of the former with the greater flexibility of the latter.

Finally we note that the fact that the scheme is with prekey, i.e., the center only has to select a hash function noninteractively once, makes management of public keys for this scheme almost identical to that of ordinary digital signature schemes.

4. Bit Commitments from Fail-Stop Signatures

The main idea in our construction of bit commitments from FSS schemes is to use the key generation protocol between signer A and center B as the commit protocol, and to think of the resulting public key as the main part of the commitment and the secret key, sk, as the string that can open the commitment.

If the FSS scheme has the almost unique secret key property, A is obviously committed to any value that can be deterministically computed from $\kappa(sk)$. There are two major difficulties, however: First, the distribution of the secret key, given the public key, is not necessarily uniform. So we need a way for A to compute a value from the secret key such that B has essentially no information about it. Here we use the universal hash functions and the extended privacy amplification result of [BBCM], cited in Theorem 2.12. Secondly, the definition of FSS schemes does *not* exclude that a cheating center can carry out key generation so that it can guess the secret key afterward. This does not contradict the security for the signer if in these cases, the signer can prove signatures made with *her own key* to be forgeries. (This, in its turn, would contradict the security for the recipient, but that property is not guaranteed if the center is cheating.) Hence, in the commitment scheme, where we want to exploit the fact that the secret key cannot be guessed, we must provide a way for the signer (now the committer) to exclude these keys.

We now describe the construction in detail. We only need an FSS scheme where one bit can be signed. Obviously, such a scheme is a special case of any more general scheme.

Protocol 4
COMMIT

1. A and B execute the key generation protocol of the FSS scheme with security parameters (k, σ), where $\sigma = 4k + 4$ and k equals the security parameter for the BC scheme we are building. Here B plays the role of the center. If A or B reject in the key generation, the commit protocol stops. Otherwise let sk be the resulting secret key and pk the public key.
2. A signs the message "0" (consisting of one 0-bit) using sk. She runs the algorithm for generating proofs of forgery on the resulting signature. If this results in a valid proof of forgery, she stops. Otherwise she continues.
3. A chooses and sends to B a random function f from a given universal$_2$ family F of hash functions with 1-bit images. The input size of f is a polynomial $p(k)$ that is an upper bound on the length of the values of $\kappa(sk)$ for the parameters k and σ. Without loss of generality, we assume that all values $\kappa(sk)$ are of precisely this length.
4. Let b be the bit A wants to commit to. Then A sends $c = f(\kappa(sk)) \oplus b$ to B, i.e., b encrypted with the privacy-amplified significant part of sk.

OPENING

1. A sends b and sk to B.
2. B verifies that $Fits(sk, pk) = 1$. He then compares b with $c \oplus f(\kappa(sk))$. If they are equal, he outputs *accept b*, if not, he outputs *reject*.

Theorem 4.1. *If the underlying FSS scheme (with the almost unique secret key property) is secure, Protocol 4 is a statistically hiding BC scheme with noninteractive opening and public verifiability.*

Proof. First note that the possibility of stopping in Step 2 of the commit protocol does not prevent two honest parties A and B from completing this protocol: the security for the recipient implies that the scheme almost never stops in Step 2 in this case. Public verifiability and noninteractive opening are clear from the construction. The fact that B will accept A's bit if both are honest follows immediately from the fact that the results of a correct key generation always satisfy the predicate *Fits*, and that f and κ are deterministic.

The *binding property* is clear from the almost unique secret key property of the FSS scheme: if a committer $\tilde{A}$ could open a commitment in two different ways, she would know two secret keys satisfying the predicate *Fits* and with different κ-images.

Hence only the *security property* remains to be shown. We need the following notation: Let *Acc* be the event that A accepts the key generation, i.e., does not stop in Step 1 of the commit protocol, and let U be the event that A does not stop in Step 1 or 2. Finally, let G be the event that the outcome of key generation, $(sk, pk, view_{\tilde{B}})$, is in the set *Good* (see Definition 2.2).

Recall that the collision entropy of a binary distribution, like that of $f(\kappa(sk))$, with

probabilities p and $1 - p$, is defined as

$$R(p) = -\log_2(p^2 + (1-p)^2)$$

and the collision information as $I = 1 - R$.

Let $I_{\tilde{B}}$ denote the collision information obtained by $\tilde{B}$ about $f(\kappa(sk))$ during the commit protocol, and let $E_{\tilde{B}}$ be its expected value, taken over the random choices of A and $\tilde{B}$. For an event X, $E_{\tilde{B}}(X)$ denotes the expected collision information given that X occurs. Then we have

$$\begin{aligned} E_{\tilde{B}} &= Prob[\neg U]E_{\tilde{B}}(\neg U) + Prob[U, G]E_{\tilde{B}}(U, G) + Prob[U, \neg G]E_{\tilde{B}}(U, \neg G) \\ &\leq 0 + Prob[U, G]E_{\tilde{B}}(U, G) + Prob[Acc, \neg G] \\ &\leq Prob[U, G]E_{\tilde{B}}(U, G) + 2^{-\sigma} \end{aligned}$$

because A does not reveal anything if the commit protocol is aborted in Step 1 or 2, and by the security for the signer (Definition 2.2).

The rest of the proof proceeds in three parts: We first show that in most cases, the best guess at the significant part of the secret key from the point of view of $\tilde{B}$ still has a rather small probability of being correct. Secondly, we use the extended privacy amplification theorem to derive that in most cases, an enemy has very little collision information about $f(\kappa(sk))$. Finally, we derive an upper bound on the average bias of the content of the commitment.

Part 1. Let SK be the random variable denoting A's secret key, and let sk_{max} denote a secret key such that $\kappa(sk_{max})$ has maximal probability given $v := (pk, view_{\tilde{B}})$, U, and G. (Thus $\kappa(sk_{max})$ is the enemy's best guess.) We now show that *on average* over the possible v's and given U and G, this maximal probability is upper bounded, i.e., on average, the best guess is not very good:

$$Prob[\kappa(SK) = \kappa(sk_{max}) \mid U, G] \leq 2^{\sigma} Prob[U, G]^{-1}. \qquad (*)$$

For this, it is sufficient to show that

$$Prob[\kappa(SK) = \kappa(sk_{max}), U, G] \leq 2^{-\sigma}.$$

To do this, we consider the following attack by a cheating center B^* on the FSS scheme.

1. B^* executes the key generation protocol with A in the same way as $\tilde{B}$ did.
2. B^* computes sk_{max} and uses it to make a signature on the message "0."

Let F denote the event that A fails to prove this forgery. Note that the distribution of the keys after Step 1 of this attack and of the commit protocol are equal. Furthermore, $U \subset Acc$, and whenever Acc and $\kappa(SK) = \kappa(sk_{\max})$ occur, U implies F by definition of the almost unique secret key property. This gives us

$$\begin{aligned} Prob[\kappa(SK) = \kappa(sk_{max}), U, G] &= Prob[\kappa(SK) = \kappa(sk_{max}), U, G, Acc] \\ &\leq Prob[F, \kappa(SK) = \kappa(sk_{max}), G, Acc] \\ &= Prob[F, Acc, \kappa(SK) = \kappa(sk_{max}) \mid G]Prob[G] \\ &\leq Prob[F \mid G] \\ &\leq 2^{-\sigma}. \end{aligned}$$

The final inequality follows from the security for the signer of the FSS scheme. This finishes the proof of (∗).

Now let V be a random variable denoting $v = (pk, view_{\tilde{B}})$ and M the set of cases where the probability that the best guess is correct is much larger than on average, and also the event that such a case occurs:

$$M := \{v \mid Prob[\kappa(SK) = \kappa(sk_{max}) \mid G, U, V = v] \geq Prob[U, G]^{-1} 2^{-\sigma/2}\}.$$

By Markov's rule, the average inequality (∗) implies that

$$Prob[M \mid U, G] \leq 2^{-\sigma/2}.$$

We split the expected information according to whether M occurs or not.

$$\begin{aligned} E_{\tilde{B}}(U, G) &\leq Prob[M \mid U, G] + Prob[\neg M \mid U, G] E_{\tilde{B}}(U, G, \neg M) \\ &\leq 2^{-\sigma/2} + \sum_{v \notin M} Prob[V = v \mid U, G] E_{\tilde{B}}(U, G, V = v). \end{aligned}$$

Part 2. Whenever $v \notin M$, the extended privacy amplification lemma, Lemma 2.13, immediately implies that the information $E_{\tilde{B}}(U, G, V = v)$ is small:

$$E_{\tilde{B}}(U, G, V = v) \leq Prob[U, G]^{-1} 2^{-\sigma/2} \frac{2}{\ln 2} < Prob[U, G]^{-1} 2^{2-\sigma/2}.$$

Substituting this into the above inequality gives

$$\begin{aligned} E_{\tilde{B}}(U, G) &< 2^{-\sigma/2} + \sum_{v \notin M} Prob[V = v \mid U, G] \, Prob[U, G]^{-1} 2^{2-\sigma/2} \\ &\leq 2^{-\sigma/2} + Prob[U, G]^{-1} 2^{2-\sigma/2} \end{aligned}$$

and thus

$$E_{\tilde{B}} \leq Prob[U, G] 2^{-\sigma/2} + 2^{2-\sigma/2} + 2^{-\sigma} < 2^{3-\sigma/2}.$$

Part 3. We now derive the bias-based security for the case $\delta = \frac{1}{2}$. In the following $\beta = bias_{1/2}(v)$ denotes $\tilde{B}$'s advantage in guessing $h(\kappa(sk))$. From the definition of the collision entropy for binary distributions, one sees

$$R(\tfrac{1}{2} + \beta) = -\log_2(\tfrac{1}{2} + 2\beta^2) = 1 - \log_2(1 + 4\beta^2) \leq 1 - 4\beta^2$$

for $|\beta| \leq \frac{1}{2}$, i.e., $4\beta^2 \leq 1$. This implies

$$\beta \leq \tfrac{1}{2}\sqrt{1 - R(\tfrac{1}{2} + \beta)}.$$

Thus we have shown the following pointwise inequality between the random variables $Bias_{1/2}$ and the collision information $I_{\tilde{B}}$:

$$Bias_{1/2} \leq \tfrac{1}{2}\sqrt{I_{\tilde{B}}}.$$

Applying the general formula $E(X) \leq \sqrt{E(X^2)}$ to $X = \sqrt{I_{\tilde{B}}}$ yields

$$E(Bias_{1/2}) \leq \tfrac{1}{2}E(\sqrt{I_{\tilde{B}}}) \leq \tfrac{1}{2}\sqrt{E(I_{\tilde{B}})} = \tfrac{1}{2}\sqrt{E_{\tilde{B}}} < 2^{1-\sigma/4}.$$

In the last inequality, the result of Part 2 was used.

As the security parameter of the FSS scheme, σ, equals $4k + 4$, this shows that the commitment scheme has the bias-based security property for $\delta = \frac{1}{2}$, and thus, by Lemma 2.7, for all δ. □

Remark. From the proof of the security property, it is clear that we do not need to hash all the way down to a 1-bit value to wipe out the enemy's information. Therefore we can commit to more than one bit in one commitment.

5. Conclusion and Open Problems

In this paper, we have shown that FSS schemes can be constructed assuming that either one-way permutations or collision-intractable hash functions exist. Both these assumptions imply the existence of (not necessarily one-to-one) one-way functions, but there is no implication known in either direction between them.

It is also unknown if FSS schemes can be constructed assuming only the existence of one-way functions, although this assumption is certainly necessary. This is an important open problem, as the existence of one-way functions is known to be sufficient for ordinary digital signatures [R], [NY].

Second, our construction from bit commitments and thus from one-way permutations is a one-time scheme. Papers about FSS schemes have often concentrated on one-time schemes, because it was known that one-time schemes and collision-intractable hash functions yield efficient schemes for many messages. Now we have the first construction from a primitive that is not known to imply such hash functions, and thus these constructions cannot be applied. It is an open problem if FSS schemes whose complexity is less than linear in the message bound N exist under these assumptions. For ordinary digital signature schemes, such constructions exist, using weaker types of hash functions, see [R].

Note, however, that our construction from hash functions does not have such problems.

Finally, it is not known if the existence of arbitrary FSS schemes is equivalent to that of FSS schemes with the almost unique secret key property, and therefore to the existence of statistically hiding BC schemes.

Acknowledgments

It is a pleasure to thank Moti Yung for interesting discussions about all sorts of statistically hiding schemes, and in particular, one that led to one of the independent discoveries of the construction of fail-stop signature schemes from bit commitments. We thank Michael Waidner for letting us extend a joint result that has never really been published. We also thank the anonymous reviewers for valuable suggestions for improvements.

Appendix. Proofs from Section 2.4

Lemma 2.7. *Let any BC scheme and any $\tilde{B}$ be given. Then for all δ:*

$$Bias_\delta \leq Bias_{1/2}.$$

Thus $\delta = \frac{1}{2}$ is the worst case on average.

Proof. It will be shown that

$$Bias_\delta = \delta(1-\delta) \sum_v |Prob[v \mid b_\delta = 0] - Prob[v \mid b_\delta = 1]|.$$

The lemma follows immediately from this, because the terms in the sum only depend on the commit protocol.

$$\begin{aligned} Bias_\delta &= \sum_v Prob[v]\, bias_\delta(v) \\ &= \sum_v |\delta\, Prob[v] - Prob[v]\, Prob[b_\delta = 0 \mid v]| \\ &= \sum_v |\delta\, Prob[v] - Prob[b_\delta = 0]\, Prob[v \mid b_\delta = 0]| \\ &= \sum_v \delta\, |Prob[v] - Prob[v \mid b_\delta = 0]|. \end{aligned}$$

Here

$$\begin{aligned} Prob[v] - Prob[v \mid b_\delta = 0] &= \delta\, Prob[v \mid b_\delta = 0] + (1-\delta)\, Prob[v \mid b_\delta = 1] \\ &\quad - Prob[v \mid b_\delta = 0] \\ &= (\delta - 1)(Prob[v \mid b_\delta = 0] - Prob[v \mid b_\delta = 1]), \end{aligned}$$

which completes the proof. □

Lemma 2.8. *The bias-based security property and the capacity-based security property of BC schemes are equivalent, except for small transformations of the security parameters. More precisely:*

1. *If a BC scheme has the bias-based security property, it has a channel capacity of at most $k2^{-k/2}$ for any $\tilde{B}$ and $k \geq 4$.*
   Thus, a capacity of at most 2^{-k} can be achieved by using a security parameter k' with $k' \geq 4$ and $k'/2 - \log_2 k' \geq k$.
2. *If a BC scheme has the capacity-based security property, it has an average bias of at most $2^{-k/2}$ for any $\tilde{B}$.*
   Thus, an average bias of at most 2^{-k} can be achieved by using the security parameter $k' = 2k$.

Proof.

Part 1. We fix $\tilde{B}$. We have to show $I(V_\delta; b_\delta) \leq k2^{-k/2}$ for all δ. We now fix δ, too. For symmetry reasons, we assume without loss of generality that $\delta \leq \frac{1}{2}$. We distinguish two cases.

Case 1: $\delta \leq 2^{-k/2}$. For such small δ's, we prove that $I(V_\delta; b_\delta)$ is small directly:

$$I(V_\delta; b_\delta) \leq H(b_\delta) = H_{bin}(\delta).$$

As H_{bin} is monotonic increasing for $\delta < \frac{1}{2}$, we can bound it by evaluating it at the maximum value of δ in this Case 1:

$$I(V_\delta; b_\delta) \leq -2^{-k/2}\left(-\frac{k}{2}\right) + (1 - 2^{-k/2})(-\log_2(1 - 2^{-k/2})).$$

As $k \geq 4$, we can use the inequality $1/(1 - 2^{-k/2}) \leq 1 + 2^{-k/2+1}$ to obtain

$$I(V_\delta; b_\delta) \leq \frac{k}{2}2^{-k/2} + \log_2(1 + 2^{-k/2+1}) \leq \frac{k}{2}2^{-k/2} + 2^{-k/2+1} \leq k2^{-k/2}.$$

Case 2: $2^{-k/2} < \delta \leq \frac{1}{2}$. Now we use the precondition that $Bias_\delta \leq 2^{-k}$. Let *Big* be the event that the individual bias is much larger, $bias_\delta(v) > 2^{-k/2-1}$. By Markov's rule, $Prob[Big] \leq 2^{-k/2+1}$. We have

$$\begin{aligned} I(V_\delta; b_\delta) &= \sum_v Prob[v](H(b_\delta) - H(b_\delta|v)) \\ &\leq Prob[Big] + \sum_{v \notin Big} Prob[v](H(b_\delta) - H(b_\delta|v)), \end{aligned}$$

because the entropy of a binary variable is at most 1.

Now we consider one fixed $v \notin Big$. Note that $H(b_\delta|v) = H_{bin}(\delta \pm bias_\delta(v))$, where $\pm$ stands for $+$ or $-$, because it is the entropy of the conditional distribution of b_δ given v. Thus we want to bound the difference of the values of H_{bin} at two closely neighboring points. We can do this if we know the maximum gradient of this function. As H'_{bin} is monotonic decreasing (see Section 2.2), its maximum is achieved at the minimum δ minus the maximum possible bias, i.e., the maximum is

$$H'_{bin}(2^{-k/2} - 2^{-k/2-1}) = -\log_2(2^{-k/2-1}) + \log_2(1 - 2^{-k/2-1}) < \frac{k}{2} + 1.$$

For $v \notin Big$ this implies

$$H(b_\delta) - H(b_\delta|v) \leq \left(\frac{k}{2} + 1\right) bias_\delta(v)$$

and finally

$$I(V_\delta; b_\delta) \leq Prob[Big] + \sum_{v \notin Big} Prob[v]\left(\frac{k}{2} + 1\right) bias_\delta(v)$$

$$\begin{aligned}
&\leq 2^{-k/2+1} + \left(\frac{k}{2}+1\right)\sum_v Prob[v] bias_\delta(v) \\
&\leq 2^{-k/2+1} + \left(\frac{k}{2}+1\right) Bias_\delta \\
&\leq 2^{-k/2+1} + \left(\frac{k}{2}+1\right) 2^{-k} \\
&\leq k2^{-k/2}.
\end{aligned}$$

This is the desired result.

Part 2. Let a BC scheme be given which satisfies the capacity-based security property. It will be shown that this scheme satisfies the bias-based security property with expected value $Bias_\delta \leq 2^{-k/2}$.

Let $\bar{B}$ be given. By Lemma 2.7 it is sufficient to show that $Bias_{1/2} \leq 2^{-k/2}$ (this will shown under the assumption that $I(V_{1/2}, b_{1/2}) \leq 2^{-k}$). The technical Lemma A.1 below shows that

$$H_{bin}(\tfrac{1}{2} - bias_{1/2}(v)) \leq 1 - 2c \cdot bias_{1/2}(v)^2,$$

where $c = (\ln 2)^{-1}$. By rewriting this and inserting $H(b_{1/2}) = 1$ we get

$$bias_{1/2}(v) \leq \sqrt{\frac{\ln 2}{2}}\sqrt{H(b_{1/2}) - H(b_{1/2} \mid v)} < \sqrt{H(b_{1/2}) - H(b_{1/2} \mid v)}.$$

Now using the formula $E(X) \leq \sqrt{E(X^2)}$ for $X = \sqrt{H(b_{1/2}) - H(b_{1/2} \mid v)}$ gives

$$\begin{aligned}
Bias_{1/2} &= \sum_v Prob[v]\, bias_{1/2}(v) \\
&\leq \sum_v Prob[v]\sqrt{H(b_{1/2}) - H(b_{1/2} \mid v)} \\
&\leq \left(\sum_v Prob[v](H(b_{1/2}) - H(b_{1/2} \mid v))\right)^{1/2} \\
&= \sqrt{I(V_{1/2}; b_{1/2})} \\
&\leq 2^{-k/2}.
\end{aligned}$$

□

Lemma A.1. Let $c^{-1} = \ln 2$. Then $H_{bin}(\frac{1}{2} - x) \leq 1 - 2cx^2$ if $|x| < \frac{1}{2}$.

Proof. The natural logarithm is given by

$$\ln(1+x) = -\sum_{i=1} \frac{(-x)^i}{i} \qquad \text{for} \quad -1 < x \leq 1.$$

Furthermore,

$$\log(\tfrac{1}{2} + x) = \log(1 + 2x) - 1 = d^{-1}(\ln(1 + 2x) - d)$$

where $d = \ln 2$ and $-\frac{1}{2} < x \leq \frac{1}{2}$. As $H_{bin}(p)$ is symmetric around $p = \frac{1}{2}$, it is sufficient to prove the lemma for $0 \leq x < \frac{1}{2}$. We get

$$\begin{aligned} dH_{bin}(\tfrac{1}{2} - x) &= -d[(\tfrac{1}{2} - x)\log(\tfrac{1}{2} - x) + (\tfrac{1}{2} + x)\log(\tfrac{1}{2} + x)] \\ &= d - (\tfrac{1}{2} - x)\ln(1 - 2x) - (\tfrac{1}{2} + x)\ln(1 + 2x) \\ &= d - \tfrac{1}{2}(\ln(1 - 2x) + \ln(1 + 2x)) + x(\ln(1 - 2x) - \ln(1 + 2x)). \end{aligned}$$

Inserting the Taylor expansion above (and remembering that x is positive) gives

$$\begin{aligned} dH_{bin}(\tfrac{1}{2} - x) &= d - \frac{1}{2}\left(\sum_{i=1}\left(\frac{-(2x)^i - (-2x)^i}{i}\right)\right) + x\left(\sum_{i=1}\left(\frac{-(2x)^i + (-2x)^i}{i}\right)\right) \\ &= d - \frac{1}{2}\left(\sum_{i\text{ even}}\left(2\frac{-(2x)^i}{i}\right)\right) + x\left(\sum_{i\text{ odd}}\left(2\frac{-(2x)^i}{i}\right)\right) \\ &= d + \left(\sum_{i\text{ even}}\frac{(2x)^i}{i}\right) - 2x\left(\sum_{i\text{ odd}}\frac{(2x)^i}{i}\right) \\ &= d + \sum_{i\text{ even}}\left(\frac{(2x)^i}{i} - \frac{(2x)^i}{i-1}\right) \\ &= d + \sum_{i\text{ even}}\frac{-(2x)^i}{i(i-1)} \\ &\leq d - \frac{(2x)^2}{2}. \end{aligned}$$

Dividing by d gives the desired result. □

Corollary 2.9. *In order to prove the capacity-based security property of a BC scheme, it is sufficient to show that* $I(V_{1/2}; b_{1/2}) \leq 2^{-k'}$, *where* k' *is such that* $k' \geq 8$ *and* $k'/4 + 1 - \log_2 k' \geq k$ (*i.e., to consider the case* $\delta = \frac{1}{2}$).

Proof. In Part 2 of the proof of Lemma 2.8, we proved the bias-based security property from the precondition $I(V_{1/2}; b_{1/2}) \leq 2^{-k}$ alone. Thus, by applying Part 1 of this lemma, we obtain the full capacity-based security property. □

References

[BBCM] C. H. Bennett, G. Brassard, C. Crépeau, and U. Maurer. Generalized privacy amplification. *IEEE Transactions on Information Theory*, **41**(6) (1995), 1915–1923.

[BBR] C. H. Bennett, G. Brassard, and J.-M. Robert. Privacy amplification by public discussion. *SIAM Journal on Computing*, **17**(2) (1988), 210–229.

[BCC] G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, **37** (1988), 156–189.

[CW] J. L. Carter and M. N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, **18** (1979), 143–154.

[D1] I. B. Damgård. Collision free hash functions and public key signature schemes. *Proceedings of Eurocrypt* '87. LNCS, 304, Springer-Verlag, Berlin, pp. 203–216, 1988.

[D2] I. B. Damgård. A design principle for hash functions. *Proceedings of Crypto '89*. LNCS, 435, Springer-Verlag, Berlin, pp. 416–427, 1990.

[DBP] H. Dobbertin, A. Bosselaers, and B. Preneel. RIPEMD-160: A strengthened version of RIPEMD. *Proceedings of the 3rd Fast Software Encryption Workshop*. LNCS, 1039, Springer-Verlag, Berlin, pp. 71–82, 1996.

[DH] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, **22**(6) (1976), 644–654.

[DPP] I. Damgård, T. P. Pedersen, and B. Pfitzmann. Statistical Secrecy and Multi-Bit Commitments. BRiCS Report Series RS-96-45. Submitted to *IEEE Transactions on Information Theory*. Available at http://www.brics.dk/

[F] W. Feller. *An Introduction to Probability Theory and its Applications*, volume II (2nd edn.). Wiley, New York, 1971.

[G] R. G. Gallager. *Information Theory and Reliable Communication*. Wiley, New York, 1968.

[GMR1] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, **17**(2) (1988), 281–308.

[GMR2] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, **18**(1) (1989), 186–208.

[HP] E. van Heyst and T. P. Pedersen. How to make efficient fail-stop signatures. *Proceedings of Eurocrypt '92*. LNCS, 658, Springer-Verlag, Berlin, pp. 366–377, 1993.

[HPP] E. van Heijst, T. P. Pedersen, and B. Pfitzmann. New constructions of fail-stop signatures and lower bounds. *Proceedings of Crypto '92*. LNCS, 740, Springer-Verlag, Berlin, pp. 15–30, 1993.

[L] L. Lamport. Constructing digital signatures from a one-way function, SRI Intl. CSL-98, Oct. 1979.

[M1] R. C. Merkle. Protocols for public key cryptosystems. In: *Secure Communications and Asymmetric Cryptosystems, AAAS Selected Symposium* 69 (G. J. Simmons, ed.). Westview Press, Boulder, CO, 1982, pp. 73–104.

[M2] R. C. Merkle. A digital signature based on a conventional encryption function. *Proceedings of Crypto '87*. LNCS, 293, Springer-Verlag, Berlin, pp. 369–378, 1988.

[NOVY] M. Naor, R. Ostrovsky, R. Venkatesan, and M. Yung. Perfect zero-knowledge arguments for NP can be based on general complexity assumptions. *Proceedings of Crypto '92*. LNCS, 740, Springer-Verlag, Berlin, pp. 196–214, 1993.

[NY] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. *Proceedings of 21st STOC*, pp. 33–43, 1989.

[P1] B. Pfitzmann. Sorting out signature schemes. *1st ACM Conference on Computer and Communications Security, Fairfax*, ACM Press, New York, pp. 74–85, 1993.

[P2] B. Pfitzmann: *Digital Signature Schemes—General Framework and Fail-Stop Signatures*. LNCS, 1100, Springer-Verlag, Berlin, August 1996.

[PP] T. P. Pedersen and B. Pfitzmann. Fail-stop signatures. To appear in *SIAM Journal on Computing*, February 1997.

[PW1] B. Pfitzmann and M. Waidner. Formal Aspects of Fail-Stop Signatures. Internal Report 22/90, Fakultät für Informatik, Universität Karlsruhe, 1990.

[PW2] B. Pfitzmann and M. Waidner. Fail-stop signatures and their application. Securicom 91, Paris, pp. 145–160.

[R] J. Rompel. One-way functions are necessary and sufficient for secure signatures. *Proceedings of 22nd STOC*, pp. 387–394, 1990.

[WP] M. Waidner and B. Pfitzmann. The dining cryptographers in the disco: unconditional sender and recipient untraceability with computationally secure serviceability. *Proceedings of Eurocrypt '89*, LNCS, 434, Springer-Verlag, Berlin, p. 690, 1990.

[W] A. D. Wyner. The wire-tap channel. *The Bell System Technical Journal*, **54**(8) (1975), 1355–1387.