Flaws in Applying Proof Methodologies to Signature Schemes

Jacques Stern[1,⋆], David Pointcheval[1], John Malone-Lee[2], and Nigel P. Smart[2]

[1] Dépt d'Informatique, ENS – CNRS, 45 rue d'Ulm, 75230 Paris Cedex 05, France
{Jacques.Stern,David.Pointcheval}@ens.fr
http://www.di.ens.fr/~{stern,pointche}
[2] Computer Science Dept, Woodland Road, University of Bristol, BS8 1UB, UK
{malone,nigel}@cs.bris.ac.uk
http://www.cs.bris.ac.uk/~{malone,nigel}

Abstract. Methods from *provable security*, developed over the last twenty years, have been recently extensively used to support emerging standards. However, the fact that proofs also need time to be validated through public discussion was somehow overlooked. This became clear when Shoup found that there was a gap in the widely believed security proof of OAEP against adaptive chosen-ciphertext attacks. We give more examples, showing that provable security is more subtle than it at first appears. Our examples are in the area of signature schemes: one is related to the security proof of ESIGN and the other two to the security proof of ECDSA. We found that the ESIGN proof does not hold in the usual model of security, but in a more restricted one. Concerning ECDSA, both examples are based on the concept of duplication: one shows how to manufacture ECDSA keys that allow for two distinct messages with identical signatures, a *duplicate signature*; the other shows that from any message-signature pair, one can derive a second signature of the same message, the *malleability*. The security proof provided by Brown [7] does not account for our first example while it surprisingly rules out malleability, thus offering a proof of a property, non-malleability, that the actual scheme does not possess.

1 Introduction

In the last twenty years *provable security* has dramatically developed, as a means to validate the design of cryptographic schemes. Today, emerging standards only receive widespread acceptance if they are supported by some form of provable argument. Of course, cryptography ultimately relies on the $\mathcal{P}$ vs. $\mathcal{NP}$ question and actual proofs are out of reach. However, various security models and assumptions allow us to interpret newly proposed schemes in terms of related mathematical results, so as to gain confidence that their underlying design is not flawed. There is however a risk that should not be underestimated: the use of provable security is more subtle than it appears, and flaws in security proofs themselves might have

⋆ The first and last examples in this paper are based on the result of an evaluation requested by the Japanese Cryptrec program and performed by this author.

M. Yung (Ed.): CRYPTO 2002, LNCS 2442, pp. 93–110, 2002.

a devastating effect on the trustworthiness of cryptography. By flaws, we do not mean plain mathematical errors but rather ambiguities or misconceptions in the security model. The first such example appeared recently, when Victor Shoup noted in [29] that there was a gap in the widely believed security proof of OAEP against adaptive chosen-ciphertext attacks. By means of a nice counter-example in a relativized model of computation, he showed that, presumably, OAEP could not be proven secure from the one-wayness of the underlying trapdoor permutation. A closer look at the literature, notably [4,2], showed that the security proof was actually valid in a weaker security model, namely against indifferent chosen-ciphertext attacks (IND-CCA1), also called lunchtime attacks [18], and *not* in the full (IND-CCA2) adaptive setting [24]. This came as a shock, even though Fujisaki, Okamoto, Pointcheval and Stern [12] were quickly able to establish that the security of RSA–OAEP could actually be proven under the RSA assumption alone, in the random oracle model. Since the more general result could not hold, a different argument based on specific properties of the RSA function had to be used.

Goldwasser, Micali and Rivest [14] introduced the notion of *existential forgery against adaptive chosen-message attacks* for public key signature schemes. This notion has become the *de facto* security definition for digital signature algorithms, against which all new signature algorithms are measured. The definition involves a game in which the adversary is given a target user's public key and is asked to produce a valid message/signature pair with respect to this public key. The adversary is given access to an oracle which will produce signatures on messages of his choice. However, the above definition does not directly deal with the most important property of a digital signature, namely *non-repudiation*: the signer should be unable to repudiate his signature. One should not that an adversary against the non-repudiation property of a signature scheme would be the legitimate signer himself. Hence, such an adversary has access to the private key, and may even control the key generation process.

The present paper gives further examples of flaws in security proofs, related to signature schemes. Two of them stem from a subtle point that has apparently been somehow overlooked: in non deterministic signature schemes, several signatures may correspond to a given message. Accordingly, the security model should unambiguously decide whether an adaptive attacker is allowed to query several signatures of the same message. Similarly, it should make clear whether obtaining a second signature of a given message, different from a previously obtained signature of the same message, is a forgery or not, and namely an existential forgery.

The first example that we give is related to the security proof offered in [22] for the ESIGN signature scheme. Crosschecking the proof, with the above observations in mind, it can be seen that it *implicitly* assumes that the attacker is not allowed to query the same message twice. Thus, the security proof does not provide security against existential forgeries under adaptive chosen-message attacks. It only applies to a more restricted class, which may be termed *single-occurrence* chosen-message attacks.

The two other examples are related to the elliptic curve digital signature algorithm ECDSA [1]. In [7], Brown uses the so-called *generic group model* to prove the security of the generic DSA, a natural analog of DSA and ECDSA in this setting. This result is viewed as supporting the security of the actual ECDSA: in the generic model, ECDSA prevents existential forgeries under adaptive chosen-message attacks. But as already remarked, this security notion does not deal with the important non-repudiation property that signature schemes should guarantee. The obvious definition is that it should be hard for a legitimate signer to produce two messages which have the same signature with respect to the same public key. If a signature scheme did not have this property then a user could publish the signature on one message and then claim it was actually the signature on another. Such a signature we shall call a *duplicate signature*, since it is the signature on two messages. This shows an inadequacy between the classical security notions and the practical requirements. Furthermore, we show that with ECDSA a signer which controls the key generation process can easily manufacture duplicate signatures, without finding a collision in the hash function. Luckily, however, our construction of duplicate signatures means that, as soon as the signer reveals the second message, the signer's private key is revealed. Concerning the generic group model, which was the sole assumption on which relies the security result provided in [7], carefully crosschecking the proof, with the above observations in mind, we see that it actually prevents a forgery which creates a different signature to a previously obtained signature of the same message. Hence, the proof implies the scheme produces non-malleable signatures. Unfortunately, ECDSA does *not* withstand such forgeries. What goes wrong here is the adequacy of the model. The proof is correct but the underlying model is flawed, since it disallows production of malleable signatures.

Note that we have not broken any of the two schemes. In particular, there are some easy ways of revising ESIGN so that it satisfies the classical security notions (see e.g. [15]).

2 Digital Signature Schemes and Security Proofs

2.1 Formal Framework

In modern terms (see [14]), a digital signature scheme consists of three algorithms $(\mathcal{K}, \Sigma, V)$:

- A *key generation algorithm* $\mathcal{K}$, which, on input 1^k, where k is the security parameter, outputs a pair $(\mathsf{pk}, \mathsf{sk})$ of matching public and private keys. Algorithm $\mathcal{K}$ is probabilistic.
- A *signing algorithm* Σ, which receives a message m and the private key sk, and outputs a signature $\sigma = \Sigma_{\mathsf{sk}}(m)$. The signing algorithm might be probabilistic.
- A *verification algorithm* V which receives a candidate signature σ, a message m and a public key pk, and returns an answer $V_{\mathsf{pk}}(m, \sigma)$ as to whether or not σ is a valid signature of m with respect to pk. In general, the verification algorithm need not be probabilistic.

Attacks against signature schemes can be classified according to the goals of the adversary and to the resources that it can use. The goals are diverse:

- Disclosing the private key of the signer. This is the most drastic attack. It is termed *total break*.
- Constructing an efficient algorithm which is able to sign any message with significant probability of success. This is called *universal forgery*.
- Providing a single message/signature pair. This is called *existential forgery*.

In terms of resources, the setting can also vary. We focus on two specific attacks against signature schemes: the *no-message attacks* and the *known-message attacks*. In the first scenario, the attacker only knows the public key of the signer. In the second, the attacker has access to a list of valid message/signature pairs. Again, many sub-cases appear, depending on how the adversary gains knowledge. The strongest is the *adaptive chosen-message attack* (CMA), where the attacker can require the signer to sign any message of its choice, where the queries are based upon previously obtained answers. When signature generation is not deterministic, there may be several signatures corresponding to a given message. A slightly weaker security model, which we call *single-occurrence adaptive chosen-message attack* (SO-CMA), allows the adversary at most one signature query for each message. In other words the adversary cannot submit the same message twice for signature.

In known-message attacks, one should point out that existential forgery becomes the ability to forge a fresh message/signature pair that has not been obtained during the attack. Again there is a subtle point here, related to the context where several signatures may correspond to a given message. We actually adopt the stronger rule that the attacker needs to forge the signature of message, whose signature was not queried. The more liberal rule, which makes the attacker successful, when it outputs a second signature of a given message, different from a previously obtained signature of the same message, will be called *malleability*.

Conversely, the *non-repudiation* property means the impossibility to produce two messages with the same signature, which will be called a *duplicate signature*. However, one should note that the adversary for such a forgery is the signer himself, who may furthermore have control on the key generation process. Such a security notion is not covered by the usual notions, and should be studied independently.

2.2 The Random Oracle Model

Ideally, one would like to obtain provable security for a signature scheme, based on the sole assumption that some underlying computational problem is hard. Unfortunately, very few schemes are currently known that allow such a proof.

The next step is to hope for a proof in a non-standard computational model, as proposed by Bellare and Rogaway [3], following an earlier suggestion by Fiat and Shamir [11]. In this model, called the random oracle model, concrete objects

such as hash functions are treated as random objects. This allows one to carry through the usual reduction arguments to the context of relativized computations, where the hash function is treated as an oracle returning a random answer for each new query. A reduction still uses an adversary as a subroutine of a program that contradicts a mathematical assumption, such as the assumption that RSA is one-way [25]. However, probabilities are taken not only over coin tosses but also over the random oracle.

Of course, the significance of proofs carried in the random oracle is debatable. Hash functions are deterministic and therefore do not return random answers. Along those lines, Canetti *et al.* [8] gave an example of a signature scheme which is secure in the random oracle model, but insecure under any instantiation of the random oracle. Despite these restrictions, the random oracle model has proved extremely useful to analyze many encryption and signature schemes. It clearly provides an overall guarantee that a scheme is not flawed, based on the intuition that an attacker would be forced to use the hash function in a non generic way.

2.3 Generic Algorithms

Recently, several authors have proposed to use yet another model to argue in favor of the security of cryptographic schemes, that could not be tackled by the random oracle model. This is the so-called *black-box* group model, or *generic* model [27,7,17]. In particular, paper [7] considered the security of ECDSA in this model. Generic algorithms had been earlier introduced by Nechaev and Shoup [19,28] to encompass group algorithms that do not exploit any special property of the encodings of group elements other than the property that each group element is encoded by a unique string. Typically, algorithms like Pollard's ρ algorithm [23] fall under the scope of this formalism, while index-calculus methods do not.

We will now go into a bit more detail of proofs in this generic model, because in one of our examples, this model is the origin of the apparent paradox. More precisely, we will focus on groups which are isomorphic to $(\mathbb{Z}_q, +)$, where q is a prime. Such groups will be called *standard cyclic groups*. An encoding of a standard cyclic group Γ is an injective map from Γ into a set of bit-strings S. We give an example: consider a subgroup of prime order of the group of points of a non-singular elliptic curve E over a finite field $\mathbb{F}$. Given a generator $\mathbf{g}$ of E, an encoding is obtained by computing $\sigma(x) = x \cdot \mathbf{g}$, where $x \cdot \mathbf{g}$ denotes the scalar multiplication of $\mathbf{g}$ by the integer x and providing coordinates for $\sigma(x)$. Note that the encoding set appears much larger than the group size, but compact encodings using only one coordinate and a sign bit ± 1 exist and, for such encodings, the image of σ is included in the binary expansions of integers $< tq$ for some small integer t, provided that q is close enough to the size of the underlying field $\mathbb{F}$. This is exactly what is recommended for cryptographic applications [16,9].

A *generic* algorithm $\mathcal{A}$ over a standard cyclic group Γ is a probabilistic algorithm that takes as input an *encoding list* $\mathcal{L} = \{\sigma(x_1), \ldots, \sigma(x_k)\}$, where each x_i is in Γ. While it executes, the algorithm may consult an oracle for further

encodings. Oracle calls consist of triples $\{i, j, \epsilon\}$, where i and j are indices of the encoding list $\mathcal{L}$ and ϵ is $\pm$. The oracle returns the string $\sigma(x_i \pm x_j)$, according to the value of ϵ and this bit-string is appended to the list $\mathcal{L}$, unless it was already present. In other words, $\mathcal{A}$ cannot access an element of Γ directly but only through its name $\sigma(x)$ and the oracle provides names for the sum or difference of two elements addressed by their respective names. Note however that $\mathcal{A}$ may access the list $\mathcal{L}$ at any time. In many cases, $\mathcal{A}$ takes as input a pair $\{\sigma(1), \sigma(x)\}$. Probabilities related to such algorithms are computed with respect to the internal coin tosses of $\mathcal{A}$ as well as the random choices of σ and x.

In [7], the adversary is furthermore allowed to include additional elements z_i' in the encoding list $\mathcal{L}$, without calling the oracle. This is consistent with the fact that one may detect whether an element is in the group or not (e.g. whether the coordinates of a point satisfy the equation which defines the elliptic curve.) However, this definitely enlarges the class of generic algorithm, compared to [19,28]. One can keep the number of additional elements smaller than twice the number of queries, since additional elements not appearing in a further query can be deleted and since each query involves at most two additional elements. Some useful results about the generic model are provided in Appendix A.1.

Again, from a methodological point of view, proofs in the generic model have to be handled with care. A specific group is not generic and specific encodings may further contradict genericity. If it happens, the exact meaning of a security proof may become highly questionable.

3 The Provable Security of ESIGN

3.1 Description of ESIGN

We follow [22], where a specification of ESIGN appears. The key generation algorithm of ESIGN chooses two large primes p, q of equal size k and computes the modulus $n = p^2q$. The sizes of p, q are set in such a way that the binary length $|n|$ of n equals $3k$. Additionally, an exponent $e > 4$ prime to $\varphi(n)$ is chosen.

Signature generation is performed as follows, using a hash function $\mathcal{H}$, outputting strings of length $k - 1$.

1. Pick at random r in $\mathbb{Z}_{pq}^{\star}$.
2. Convert $(0\|\mathcal{H}(m)\|0^{2k})$ into an integer y and compute $z = (y - r^e) \bmod n$.
3. Compute $w_0 = \lceil z/pq \rceil$ and $w_1 = w_0.pq - z$. If $w_1 \geq 2^{2k-1}$, return to step 1.
4. Set $u = w_0 \cdot (er^{e-1})^{-1} \bmod p$ and $s = r + upq$.
5. Output s as the signature of m.

The basic paradigm of ESIGN is that the arithmetical progression $r^e \bmod n + tpq$ consists of e-th powers of easily computed integers: one adjusts t so as to fall into a prescribed interval of length 2^{2k-1}.

Signature verification converts integer $s^e \bmod n$ into a bit string S of length $3k$ and checks that $[S]^k = 0\|\mathcal{H}(m)$, where $[S]^k$ denotes the k leading bits of S.

3.2 The Approximate e-th Root Problem

As noted in the previous section, RSA moduli of the from p^2q offer a very efficient way to solve the following problem, having knowledge of the factorization of n: given n and y in $\mathbb{Z}_n^\star$, find x such that $x^e \bmod n$ lies in the interval $[y, y+2^{2k-1})$, where the bit-size of n is $3k$ and $[y, y+2^{2k-1})$ denotes $\{u | y \leq u < y+2^{2k-1}\}$.

It is conjectured that the above problem, called the approximate e-th root problem (AERP) in [22], is hard to solve. More precisely, denote by $\mathsf{Succ}^{\mathsf{aerp}}(\tau, k)$ the probability for any adversary $\mathcal{A}$ to find an element whose e-th power lies in the prescribed interval, within time τ. In symbols, it reads

$$\Pr[(n,e) \leftarrow \mathcal{K}(1^k), y \leftarrow \mathbb{Z}_n, x \leftarrow \mathcal{A}(n,e,y) : (x^e \bmod n) \in [y, y+2^{2k-1})],$$

then, for large enough moduli, this probability is extremely small. Variants of the above can be considered, where the length of the interval is replaced by 2^{2k} or 2^{2k+1}.

Of course, the factorization of n allows to solve the AERP problem. It is unknown whether the converse is true, i.e. whether AERP and inverting RSA are computationally equivalent. Various attacks against AERP are known for $e = 2, 3$ (see [5,30]). However, it is fair to say that there is no known attack against AERP when e is greater or equal than 4.

3.3 The Security Proof

For this signature scheme, one can prove, in the random oracle model, the following security result, where $T_{exp}(k)$ denotes the computing time of modular exponentiation modulo a $3k$-bit integer.

Theorem 1. *Let $\mathcal{A}$ be a* SO-CMA*-adversary against the ESIGN signature scheme that produces an existential forgery, with success probability ε, within time τ, making q_H queries to the hash function and q_s distinct requests to the signing oracle respectively. Then, AERP can be solved with probability ε', and within time τ', where*

$$\varepsilon' \geq \frac{\varepsilon}{q_H} - (q_H + q_s) \times (3/4)^k - \frac{1}{2^{k-1}} \quad \textit{and} \quad \tau' \leq \tau + k(q_s + q_H) \cdot T_{exp}(k).$$

Our method of proof is inspired by Shoup [29] and differs from [22]: we define a sequence of Game_1, Game_2, etc of modified attack games starting from the actual game Game_0. Each of the games operates on the same underlying probability space, only the rules defining how the view is computed differ from game to game.

Proof. (of Theorem 1). We consider an adversary $\mathcal{A}$ outputting an existential forgery (m, s), with probability ε, within time τ. We denote by q_H and q_s respectively the number of queries from the random oracle $\mathcal{H}$ and from the signing oracle. As explained, we start by playing the game coming from the actual adversary, and modify it step by step, until we reach a final game, whose success probability has an upper-bound obviously related to solving AERP.

Game_0**:** The key generation algorithm $\mathcal{K}(1^k)$ is run and produces a pair of keys $(\mathsf{pk}, \mathsf{sk})$. The adversary $\mathcal{A}$ is fed with pk and, querying the random oracle $\mathcal{H}$ and the signing oracle Σ_{sk}, it outputs a pair (m, s). We denote by S_0 the event that $V_{\mathsf{pk}}(m, s) = 1$. We use a similar notation S_i in any Game_i below. By definition, we have $\Pr[S_0] = \varepsilon$.

Game_1**:** In this game, we discard executions, which end up outputting a valid message/signature pair (m, s), such that m has not been queried from $\mathcal{H}$. This means restricting to the event AskH that m has been queried from $\mathcal{H}$. Unwinding the ESIGN format, we write: $s^e = 0 \,\|\, w \,\|\, \star \bmod n$. If AskH does not hold, $\mathcal{H}(m)$ is undefined, and the probability that $\mathcal{H}(m) = w$ holds is $1/2^{k-1}$: $\Pr[S_0 \,|\, \neg\mathsf{AskH}] \leq 2^{-k+1}$. Thus, $\Pr[S_1] = \Pr[S_0 \wedge \mathsf{AskH}] \geq \Pr[S_0] - 2^{-k+1}$.

Game_2**:** In this game, we choose at random an index κ between 1 and q_H. We let m_κ be the κ-th message queried to $\mathcal{H}$. We then discard executions which output a valid message/signature pair (m, s), such that $m \neq m_\kappa$. Since the additional random value κ is chosen independently of the execution of Game_1, $\Pr[S_2] = \Pr[S_1]/q_H$.

Game_3**:** In this game, we immediately abort if a signing query involves message m_κ. By the definition of existential forgery, this only eliminates executions outside S_2. Thus: $\Pr[S_3] = \Pr[S_2]$.

Game_4**:** We now simulate the random oracle $\mathcal{H}$, by maintaining an appropriate list, which we denote by $\mathsf{H\text{-}List}$. For any fresh query m, we pick at random $u \in \mathbb{Z}_n$ and compute $z = u^e \bmod n$, until the most significant bit of z is 0. We next parse z as $0 \,\|\, w \,\|\, \star$, where w is of length $k - 1$ and check whether $z - w.2^{2k}$ is less than 2^{2k-1}. If this is true, we store (m, u, w) in $\mathsf{H\text{-}List}$ and returns w as the answer to the oracle call. Otherwise we restart the simulation of the current query. However, we stop and abort the game after k trials. This game differs from the previous one if z remains undefined after k attempts: $|\Pr[S_4] - \Pr[S_3]| \leq (q_H + q_s) \times (3/4)^k$.

Game_5**:** We modify the simulation by replacing $\mathcal{H}(m_\kappa)$ by v, where v is a bit string of length $k - 1$, which serves as an additional input. The distribution of $\mathcal{H}$-outputs is unchanged: $\Pr[S_5] = \Pr[S_4]$.

Game_6**:** We finally simulate the signing oracle: for any m, whose signature is queried, we know that $m \neq m_\kappa$ cannot hold, since corresponding executions have been aborted. Thus $\mathsf{H\text{-}List}$ includes a triple (m, u, w), such that $u^e \bmod n$ has its k leading bits of the form $0\|\mathcal{H}(m)$. Accordingly, u provides a valid signature of m. Therefore, $\Pr[S_6] = \Pr[S_5]$.

Summing up the above inequalities, we obtain

$$\Pr[S_6] \geq \Pr[S_3] - (q_H + q_s) \times (3/4)^k \geq \frac{\varepsilon}{q_H} - (q_H + q_s) \times (3/4)^k - \frac{1}{2^{k-1}}.$$

When Game_6 terminates outputting a valid message/signature pair (m, s), we unwind the ESIGN format and get $s^e = (0 \,\|\, v \,\|\, \star) \bmod n$, with $v = \mathcal{H}(m)$. If S_6 holds, we know that $m = m_\kappa$ and $\mathcal{H}(m) = v$. This leads to an element whose e-th power lies in the interval $[v2^{2k}, v2^{2k} + 2^{2k})$, thus solving an instance of AERP.

We finally have: $\Pr[S_6] \leq \mathsf{Succ}^{\mathsf{aerp}}(\tau', k)$, where τ' denotes the running time of Game_6. This is the requested bound. Observe that τ' is the sum of the time for the original attack, plus the time required for simulations, which amounts to at most $k(q_s + q_H)$ modular exponentiations. We get $\tau' \leq \tau + k(q_s + q_H) \cdot T_{exp}(k)$. □

3.4 Comments on the Security Model

We definitely had to use the SO-CMA model. If the adversary was allowed to submit the same message twice to the signing oracle, the simulation would fail at the second call, since there is a single signature available. Thus, contrarily to what is claimed in [22], the result only applies to single-occurrence adaptive chosen-message attacks. We do not know how to extend the proof to deal with the stronger CMA model.

4 Duplicates in ECDSA

Let us now turn to the ECDSA signature scheme, on which we give two more examples.

4.1 Description of ECDSA

The ElGamal signature scheme [10] appeared in 1985 as the first DL-based signature scheme. In 1989, using the Fiat and Shamir heuristic [11] based on fair zero-knowledge [13], Schnorr provided a zero-knowledge identification scheme [26], together with the corresponding signature scheme. In 1994, a digital signature standard DSA [20] was proposed, whose flavor was a mixture of ElGamal and Schnorr. The standard was later adapted to the elliptic curve setting under the name ECDSA [1,20]. Following [6,7], we propose the description of a generic DSA (see Figure 1), which operates in any cyclic group $\mathcal{G}$ of prime order q, thanks to a reduction function. This reduction function f applies to any element of the group $\mathcal{G}$, into $\mathbb{Z}_q$. In the DSA, f takes as input an integer modulo p and outputs $f(r) = r \bmod q$. In the elliptic curve version [1,20,9], the function is defined in

Initialization
$\mathcal{G}$ a cyclic group of prime order q
$\mathbf{g}$ a generator of $\mathcal{G}$
$H : \{0,1\}^\star \to \{0,1\}^h$ a hash function
$f : \mathcal{G} \to \mathbb{Z}_q$ a reduction function

$\mathcal{K}$**: Key Generation** $\to (\mathbf{y}, x)$
private key $\quad 0 < x < q$
public key $\quad \mathbf{y} = x \cdot \mathbf{g}$

Σ**: Signature of** $m \to (r, s)$
k randomly chosen $0 < k < q$
$\mathbf{r} = k \cdot \mathbf{g} \quad r = f(\mathbf{r})$
if $r = 0$ abort and start again
$e = H(m) \quad s = k^{-1}(e + xr) \bmod q$
if $s = 0$ abort and start again

V**: Verification of** $(m, r, s) \to$ `valid ?`
check whether $0 < r, s < q$ and $r = f(\mathbf{r}')$
where $e = H(m)$ and $\mathbf{r}' = es^{-1} \cdot \mathbf{g} + rs^{-1} \cdot \mathbf{y}$

Fig. 1. The Generic DSA

a more intricate manner, which we now describe. An elliptic curve point $\mathbf{r}$ is given by two coordinates (x, y), which take values in the base field. For elliptic curves over prime fields, one simply sets $f(\mathbf{r}) = x \bmod q$. For curves over $\mathbb{F}_{2^m}$, x is a sequence of m bits and $f(\mathbf{r})$ is obtained by first turning x into an integer less than 2^m, by a standard conversion routine. Anyway, one just has to keep in mind that in ECDSA the function f depends on the x-coordinate only, and thus $f(-\mathbf{r}) = f(\mathbf{r})$.

Before we review the security results proven about ECDSA, namely in [7], let us show some surprising properties of the scheme due to the above choice of reduction function f.

4.2 Duplicate Signatures

Let us first describe how to produce duplicate signatures for ECDSA. Recall we have two messages m_1 and m_2 and we wish to produce a signature which is valid for both messages, with a possible control on the key generation process. We will do this by "concocting" a public/private key pair, hence we see that our method assumes that the two target messages are known to the signer before he generates his public/private key pair. We note that the special key pair is still valid and the user is still able to sign other messages as usual.

We first compute $h_1 = H(m_1)$ and $h_2 = H(m_2)$. We generate a random $k \in \{1, \ldots q-1\}$, compute $r = f(k \cdot \mathbf{g})$, and then set the private key to be

$$x = -\left(\frac{h_1 + h_2}{2r}\right) \bmod q,$$

with the public key being given by $\mathbf{y} = x \cdot \mathbf{g}$. To generate our duplicate signature on m_1 and m_2 we compute $s = k^{-1}(h_1 + xr) \bmod q$.

That (r, s) is a valid signature on m_1 follows from the definition of ECDSA, we only need to show that (r, s) is also a valid signature on m_2. We evaluate the $\mathbf{r}'$ in the verification algorithm for the signature (r, s) on the message m_2, noting that $rx = -(h_1 + h_2)/2 \bmod q$,

$$\mathbf{r}' = (h_2/s)\mathbf{g} + (r/s)\mathbf{y} = \left(\frac{h_2 + rx}{s}\right)\mathbf{g} = k\left(\frac{h_2 - h_1}{h_1 - h_2}\right)\mathbf{g} = -k \cdot \mathbf{g} = -\mathbf{r}.$$

Hence, $f(\mathbf{r}') = f(-\mathbf{r}) = f(\mathbf{r}) = r$ and the signature verifies.

Example. As an example we use one of the recommended curves from X9.62 [1]. The curve is defined over $\mathbb{F}_p$ where $p = 2^{192} - 2^{64} - 1$, and is given by equation $y^2 = x^3 - 3x + b$, where

$$b = \texttt{0x64210519E59C80E70FA7E9AB72243049FEB8DEECC146B9B1}.$$

This curve has prime group order given by

$$q = 6277101735386680763835789423176059013767194773182842284081,$$

and a base point is given by $\mathbf{g} = (X, Y)$ where

$$X = \texttt{0x188DA80EB03090F67CBF20EB43A18800F4FF0AFD82FF1012},$$
$$Y = \texttt{0x07192B95FFC8DA78631011ED6B24CDD573F977A11E794811}.$$

Suppose we have a public key given by $\mathbf{y} = (X', Y')$

$$X' = \texttt{0xA284DB03CAC23298DF9FD9C60560B16292FBE5C7E2C26C25},$$
$$Y' = \texttt{0x3F9EABD65A25DA6E72285670AA3D639B381952AFDDECEBAA}.$$

Consider the two, hundred byte messages $m_1 = [0, 1, 2, 3, \ldots, 99]$ and $m_2 = [10, 11, 12, 13, \ldots, 109]$, with hash values, computed via SHA-1 [21],

$$h_1 = \mathsf{SHA\text{-}1}(m_1) = \texttt{0x1E6634BFAEBC0348298105923D0F26E47AA33FF5},$$
$$h_2 = \mathsf{SHA\text{-}1}(m_2) = \texttt{0x71DDBA9666E28406506F839DAA4ECAF8D03D2440}.$$

A duplicate signature on both m_1 and m_2 is provided by (r, s), with

$$r = \texttt{0x7B3281ED9C01372E09271667D88F840BEB888F43AF4A7783},$$
$$s = \texttt{0xAFC81CEC549C77F00B4790160A584FD636BB049FD9D9E0BD}.$$

Note that, as soon as one publishes the duplicate signature, a third party can recover the signer's private key and so is able to forge messages. Hence, this example of duplicate signatures should not be considered a security weakness. However, one does not know that no other duplicate signature exists, for this or any other signature scheme, which do not arise from collisions in the hash function.

4.3 Malleability

Still using the above specific property of f, that is $f(-\mathbf{r}) = f(\mathbf{r})$, ECDSA is easily malleable. Indeed, from a signature (r, s) of a message m, whatever the keys are, one can derive a second signature, namely $(r, -s)$. Referring to Figure 1, we see that the values of $\mathbf{r}'$ that appear in the verification of both signatures are symmetric, so that their image by f is the same.

4.4 Comments on the Security Results

Let us now see whether the above security notions have been appropriately dealt with or not in the security analyses which appeared in the literature. For the reader's convenience, we include in Appendix A.3 our own version of the theorem and its proof (it is highly based on [7]). The original theorem [7] claims that the generic DSA withstands existential forgeries against adaptive chosen-message attacks, in the generic model, under some assumptions, namely the collision-resistance of the hash function and the *almost-invertibility* of the reduction function (see more details in Appendix A.2). This latter property is not satisfied for DSA, but is clearly satisfied with the reduction function used in ECDSA: given an x-value, if it does not correspond to the x-coordinate of a point on the curve, g outputs Fail, otherwise it randomly outputs one of the

(two) corresponding points. Hasse's theorem ensures that g is an almost inverse of f. It furthermore helps to say that the statistical distance between $\mathcal{D}_g$ and $\mathcal{U}$ is less than $5/q$. Therefore, f is $(5/q, t)$-almost-invertible for any t.

Going through the proof, the reader can check that it actually establishes that, in the *generic model*, ECDSA is non-malleable under the collision-resistance of the hash function only. The question now becomes: what is the meaning of a proof supporting a scheme by means of an ideal model where the scheme has a property (non-malleability) that it does not have in reality? The flaw here comes from the encoding which is not generic because of the automorphism. Notice that Koblitz curves, as advocated in some standards, are even "less" generic since they have more automorphisms.

About the duplicate signatures, the proof does not deal with the problem at all, since as already remarked, for non-repudiation the adversary is the signer himself. The methodological lesson is that in some scenarios non-repudiation does not necessarily follows from resistance to existential forgeries. In other words, the security model does not properly account on a possible collusion between the key generation algorithm and the signing algorithm. Whilst our example of duplicate signatures is not a security concern, there may be others. Hence, the proof methodology and security model should allow for this.

5 Conclusion

We have shown that the version of the ESIGN cryptosystem described in the P1363 submission [22] withstands existential forgery against single-occurrence adaptive chosen-message attacks, based on the hardness of AERP. However, the proof does not extend to the usual CMA scenario. We have also considered a new kind of attack, independent of existential forgeries, since the attacker may be the signer himself. We have illustrated it on ECDSA. It shows that non-repudiation is not totally encompassed by usual security analyses. Finally, we have proved the non-malleability of the generic DSA under adaptive chosen-message attacks. This is in contrast with the actual malleability of ECDSA and puts some doubts on the significance of the generic model.

In conclusion, we give the warning to practitioners, that security proofs need some time to be discussed, accepted, and interpreted within the research community.

Acknowledgments

We thank Tatsuaki Okamoto for fruitful discussions. It should be emphasized that the first and the last examples are based on the result of an evaluation requested by the Japanese Cryptrec program and performed by the first named author. This author wishes to thank Cryptrec.

References

1. American National Standards Institute. Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm. ANSI X9.62-1998, January 1999.

2. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among Notions of Security for Public-Key Encryption Schemes. In *Crypto '98*, LNCS 1462, pages 26–45, Springer-Verlag, 1998.
3. M. Bellare and P. Rogaway. Random Oracles Are Practical: a Paradigm for Designing Efficient Protocols. In *Proc. of the 1st CCS*, pages 62–73, ACM Press, 1993.
4. M. Bellare and P. Rogaway. Optimal Asymmetric Encryption – How to Encrypt with RSA. In *Eurocrypt '94*, LNCS 950, pages 92–111, Springer-Verlag, 1995.
5. E. Brickell and J. M. DeLaurentis. An Attack on a Signature Scheme proposed by Okamoto and Shiraishi. In *Crypto '85*, LNCS 218, pages 28–32, Springer-Verlag, 1986.
6. E. Brickell, D. Pointcheval, S. Vaudenay, and M. Yung. Design Validations for Discrete Logarithm Based Signature Schemes. In *PKC '2000*, LNCS 1751, pages 276–292, Springer-Verlag, 2000.
7. D. R. L. Brown. The Exact Security of ECDSA, January 2001. IEEE 1363 [16].
8. R. Canetti, O. Goldreich, and S. Halevi. The Random Oracles Methodology, Revisited. In *Proc. of the 30th STOC*, pages 209–218, ACM Press, 1998.
9. Certicom. Standards for efficient cryptography, September 2000.
10. T. ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, IT–31(4):469–472, July 1985.
11. A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions of Identification and Signature Problems. In *Crypto '86*, LNCS 263, pages 186–194, Springer-Verlag, 1987.
12. E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA–OAEP is Secure under the RSA Assumption. In *Crypto '2001*, LNCS 2139, pages 260–274, Springer-Verlag, 2001.
13. S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. In *Proc. of the 17th STOC*, pages 291–304, ACM Press, 1985.
14. S. Goldwasser, S. Micali, and R. Rivest. A Digital Signature Scheme Secure Against Adaptative Chosen-Message Attacks. *SIAM Journal of Computing*, 17(2):281–308, April 1988.
15. L. Granboulan. How to repair ESIGN. NESSIE internal document, may 2002. See `http://www.cryptonessie.org/`. Document NES/DOC/ENS/WP5/019.
16. IEEE P1363. Standard Specifications for Public Key Cryptography, August 1998. See `http://grouper.ieee.org/groups/1363/`.
17. D. Naccache, D. Pointcheval, and J. Stern. Twin Signatures: an Alternative to the Hash-and-Sign Paradigm. In *Proc. of the 8th CCS*, ACM Press, 2001.
18. M. Naor and M. Yung. Public-Key Cryptosystems Provably Secure against Chosen Ciphertext Attacks. In *Proc. of the 22nd STOC*, pages 427–437. ACM Press, 1990.
19. V. I. Nechaev. Complexity of a Determinate Algorithm for the Discrete Logarithm. *Mathematical Notes*, 55(2):165–172, 1994.
20. NIST. Digital Signature Standard (DSS). Federal Information Processing Standards Publication 186, November 1994. Revision (To include ECDSA) : 186-2, January 2000.
21. NIST. Secure Hash Standard (SHS). Federal Information Processing Standards Publication 180–1, April 1995.
22. T. Okamoto, E. Fujisaki and H. Morita. TSH-ESIGN: Efficient Digital Signature Scheme Using Trisection Size Hash, 1998. IEEE 1363 [16].
23. J. M. Pollard. Monte Carlo Methods for Index Computation (mod p). *Mathematics of Computation*, 32(143):918–924, July 1978.

24. C. Rackoff and D. R. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In *Crypto '91*, LNCS 576, pages 433–444. Springer-Verlag, 1992.
25. R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.
26. C. P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3):161–174, 1991.
27. C. P. Schnorr and M. Jakobsson. Security of Signed ElGamal Encryption. In *Asiacrypt '2000*, LNCS 1976, pages 458–469, Springer-Verlag, 2000.
28. V. Shoup. Lower Bounds for Discrete Logarithms and Related Problems. In *Eurocrypt '97*, LNCS 1233, pages 256–266, Springer-Verlag, 1997.
29. V. Shoup. OAEP Reconsidered. In *Crypto '2001*, LNCS 2139, pages 239–259, Springer-Verlag, 2001.
30. B. Vallée, M. Girault and P. Toffin. How to break Okamoto's Cryptosystem by Reducing Lattice Bases. In *Eurocrypt '88*, LNCS 330, pages 281–292, Springer-Verlag, 1988.

A The Security Proof of ECDSA

A.1 Proofs in the Generic Model

With the proofs in the generic model, we identify the underlying probabilistic space with the space $S^{n+2} \times \Gamma \times \Gamma^{2n}$, where S is the set of bit-string encodings. Given a tuple $\{z_1, \ldots, z_{n+2}, x, x_1, \ldots, x_{2n}\}$ in this space, z_1 and z_2 are used as $\sigma(1)$ and $\sigma(x)$, the successive z_i are used in sequence to answer the n oracle queries and the $x_i \in \Gamma$ serve as pre-images of the additional elements z'_i (in the group) included by the adversary into the encoding list $\mathcal{L}$. However, this interpretation may yield inconsistencies as it does not take care of possible collisions.

We give another interpretation of the encoding σ. This interpretation is based on defining from the tuple $\{z_1, \ldots, z_{n+2}\}$, a sequence of polynomials $F_i(X, X_1, \ldots, X_{2n})$, with coefficients modulo q, depending on the execution of $\mathcal{A}$:

- Polynomials F_1 and F_2 are set to $F_1 = 1$ and $F_2 = X$, respectively. Thus $\mathcal{L} = \{F_1, F_2\}$.
- When the adversary puts an additional k-th element z'_k in the encoding list, polynomial F_{n+k+2} is defined as X_k, and added to $\mathcal{L}$.
- At the ℓ-th query $\{i, j, \epsilon\}$, polynomial F_ℓ is defined as $F_i \pm F_j$, where the sign $\pm$ is chosen according to ϵ. If F_ℓ is already listed as a previous polynomial $F_h \in \mathcal{L}$, then F_ℓ is marked and $\mathcal{A}$ is fed with the answer corresponding to h. Otherwise, z_ℓ is returned by the oracle and F_ℓ is added to $\mathcal{L}$.

Observe that all F_i polynomials are *affine*, i.e. of the form $a_0 + \sum_{i=1}^{j} a_i X_i$.

Once $\mathcal{A}$ has come to a stop, variable X is set to x, and the X_ks are set to x_k. In other words, σ is set at random, subject to the conditions $z_\ell = \sigma(F_\ell(x, x_1, \ldots, x_{2n}))$, $\ell = 1, \ldots, n+2$ and $z'_k = \sigma(x_k)$, $k = 1, \ldots, 2n$. It is

easy to check that the behavior of the algorithm that is driven by the polynomials F_i is exactly similar to the behavior of the regular algorithm, granted that elements in the sequence $(z_1, \ldots, z_{n+2})$ are all distinct, and that no polynomial $F_i - F_j$ vanishes at $(x, x_1, \ldots, x_{2n})$, where i, j range over the $3n+2$ indices of polynomials in $\mathcal{L}$. We call a sequence $\{z_1, \ldots, z_{n+2}, x, x_1, \ldots, x_{2n}\}$ which satisfies both requirements a *safe* sequence. As explained, an encoding σ can be defined from a safe sequence, such that:

$$\sigma(F_i(x, x_1, \ldots, x_{2n})) = z_i, \text{ for all unmarked } F_i, \text{ and } 1 \leq i \leq n+2,$$
$$\sigma(x_k) = z'_k, \text{ for } k = 1, \ldots, 2n.$$

This correspondence preserves probabilities. However, it does not completely cover the sample space $\{\sigma, x\}$ since executions such that $F_i(x, x_1, \ldots, x_{2n}) = F_j(x, x_1, \ldots, x_{2n})$, for some indices i, j, such that F_i and F_j are not identical are omitted. The following lemmas allow to bound the probability of unsafe sequences.

Lemma 1. *Let P be a non-zero affine polynomial in $\mathbb{Z}_q[X_1, \ldots, X_j]$, then*

$$\Pr_{x_1,\ldots,x_j \in \mathbb{Z}_q}[P(x_1, \ldots, x_j) = 0] \leq \frac{1}{q}.$$

Lemma 2. *Assume $n^2 < q$. The probability of unsafe sequences is at upper-bounded by $5(n+1)^2/q$.*

Proof. We first observe that sequences of random elements $\{z_1, z_2, \ldots, z_{n+2}\}$, which are not all distinct appear with probability

$$1 - \prod_{k=1}^{n+1}\left(1 - \frac{k}{q}\right) \leq 1 - \left(1 - \sum_{k=1}^{n+1}\frac{k}{q}\right) \leq \frac{(n+1)(n+2)}{2q}.$$

Next, using Lemma 1, we can bound the probability that $F_i - F_j$ vanishes at $(x, x_1, \ldots, x_{2n})$ by $1/q$. Since there are at most $\binom{3n+2}{2}$ such polynomials, we infer that, once $\{z_1, \ldots, z_{n+2}\}$ have been set and are distinct, the set of $(x, x_1, \ldots, x_{2n})$ such that $\{z_1, \ldots, z_{n+2}, x, x_1, \ldots, x_{2n}\}$ is not safe has probability bounded by $\binom{3n+2}{2}/q = (3n+2)(3n+1)/2q$. One easily completes the proof. □

A.2 Preliminaries

Let f be a reduction function $f : \mathcal{G} \to \mathbb{Z}_q$. An almost-inverse g of f is a probabilistic algorithm g, possibly outputting Fail, such that

$$(i)\ \Pr_{b \in_R \mathbb{Z}_q}[g(b) \in \mathcal{G} \wedge f(g(b)) = b] \geq 1/3$$

Function f is (δ, t)-*almost-invertible*, with almost-inverse g, if furthermore:

$$(ii)\ \mathcal{D}_g \approx_\delta \mathcal{U}, \text{ where } \begin{cases} \mathcal{D}_g = \{g(b) \,|\, b \in_R \mathbb{Z}_q \wedge g(b) \in \mathcal{G}\} \\ \mathcal{U} = \{a \,|\, a \in_R \mathcal{G}\}. \end{cases}$$

In the second item, notation $\mathcal{D}_g \approx_\delta \mathcal{U}$ means that no distinguisher with running time bounded by t can get an advantage greater than δ.

A.3 The Security Proof

We now prove the security of the generic DSA in the generic model. We follow [7], but we adopt a different style of proof, inspired by Shoup [29]. Referring to Figure 1, we clarify our use of encodings. The base point $\mathbf{g}$ of the group $\mathcal{G}$ is identified with the canonical generator 1 of $\mathbb{Z}_q$ and therefore labeled by $\sigma(1)$. Similarly, the public key $\mathbf{y}$ is labeled by $\sigma(x)$, where x is the private key. When an element $\mathbf{r}$ is requested, at signature generation, it is obtained as $\sigma(k)$, where k is randomly chosen. Finally, the reduction function f directly operates on the set of encodings S. Contrary to the earlier approach of [27], we do not model the hash function as a random oracle. Rather, along the lines first investigated in [6], we use specific properties of the hash function, such as one-wayness or collision resistance.

A couple of lemmas will be needed. We first show how one can perfectly simulate the distribution of valid signatures. We define a simulator $\mathcal{S}$. The simulator, picks elements $u \in_R S$, and $s \in_R \mathbb{Z}_q$, and outputs the pair (r, s), with $r = f(u)$.

Lemma 3. *For any message m, the output distribution of $\mathcal{S}$ is perfectly indistinguishable from the output distribution of $\Sigma_{\mathsf{sk}}(m)$.*

We also state an easy lemma from elementary probability theory.

Lemma 4. *Let $\mathbf{S}$ be a binomial distribution, which is the sum of $k = 5 \ln n$ Bernoulli trials with probability for success $\geq 1/3$. Then, the probability that $\mathbf{S} = 0$ is at most $1/n^2$.*

We finally state the security result.

Theorem 2. *Let Γ be a standard cyclic group of prime order q. Let S be a set of bit-string encodings. Let $H : \{0,1\}^* \to \{0,1\}^h$ be a hash function and $f : S \to \mathbb{Z}_q$ be a reduction function with almost-inverse g. Let $\mathcal{A}$ be a generic algorithm over Γ, that makes at most q_s queries to the signing oracle and n queries to the group-oracle, respectively. Assume that $\mathcal{A}$, on input $\{\sigma(1), \sigma(x)\}$, returns a message m and a valid generic DSA signature (r, s) of m, achieving malleability with probability $\varepsilon = \mathsf{Succ}^{\mathsf{cma}}(\mathcal{A})$, within running time t. Then there exist adversaries $\mathcal{B}_H$, $\mathcal{C}_H$, $\mathcal{D}_g$, operating within time bound t', and such that $\mathcal{B}_H$ is attempting to invert $H' = H \bmod q$ with success probability ε_H, $\mathcal{C}_H$ is attempting to find collisions for $H' = H \bmod q$ with success probability γ_H, and $\mathcal{D}_g$ is playing a distinguishing game for g, with advantage δ_g, where*

$$\varepsilon \leq 2\gamma_H + 2n(\delta_g + \varepsilon_H) + \frac{5(n+1)(n+q_s+1)}{q},$$
$$t' \leq t + n \times (5\tau_g \ln n + \tau_H),$$

with τ_g the running time of g and τ_H the running time for H.

Proof. Let $\mathcal{A}$ be a generic attacker able to forge a pair consisting of a message m and a valid signature (r, s). We assume that, once these outputs have been issued, $\mathcal{A}$ goes on checking the signature by requesting the encoding of $es^{-1} + xrs^{-1} \bmod$

q, where $e = H(m)$, and checking that its image under f is r. The request can be performed by mimicking the usual double-and-add algorithm, calling the generic encoding at each group operation. We assume furthermore, that, after each query m_j to the signing oracle, the adversary immediately performs a similar request to check the validity of the answer. To keep things simple, we do not perform any book-keeping of the additional requests and keep n to denote the overall number of queries to the group oracle. We now play games as before:

Game_0: An encoding σ is chosen and a key pair $(\mathsf{pk}, \mathsf{sk})$ is generated using $\mathcal{K}(1^k)$. Adversary $\mathcal{A}$ is fed with pk and, querying the generic encoding and the signing oracle, outputs a message m and a signature (r, s). We denote by S_0 the event $V_{\mathsf{pk}}(m, (r, s)) = 1$ and use a similar notation S_i in any Game_i below. By definition, we have $\Pr[S_0] = \varepsilon$.

Game_1: We slightly modify this game, by using the interpretation of the encoding proposed in Section A.1: this uses a sequence $\{z_1, \ldots, z_{n+2}, x, x_1, \ldots, x_{2n}\}$. As shown in Section A.1, in Lemma 2, the new game only differs from the old on unsafe sequences: $|\Pr[S_1] - \Pr[S_0]| \leq 5(n+1)^2/q$.

Game_2: In this game, we perform additional random tests, without modifying the simulation of the generic oracle: a test is performed at each index ℓ, such that the corresponding affine polynomial appears for the first time (or is unmarked following the terminology of Section A.1). Let $F_\ell = b_\ell X + a_\ell$. We pick at random $\tilde{e_\ell} \in_R \mathbb{Z}_q$, and compute $c_\ell \leftarrow g(b_\ell a_\ell^{-1} \tilde{e_\ell} \bmod q)$ until the computation of g returns an answer different from Fail. However, we stop and abort the game after $5 \ln n$ trials. This game differs from the previous one if c_ℓ remains undefined after $5 \ln n$ attempts. Since $\tilde{e_\ell}$ is uniformly distributed, and since the successive trials are mutually independent, we may use Lemma 4 and bound the corresponding probability by $1/n^2$. This provides the overall bound $1/n$, when ℓ varies. Taking into account the fact that the experiments are independent from the execution of Game_1, we get $\Pr[S_2] \geq (1 - 1/n) \Pr[S_1]$.

Game_3: Here, we further modify the previous game by letting c_ℓ replace z_ℓ, for each index ℓ such that F_ℓ is unmarked. Note that we have $f(z_\ell) = b_\ell a_\ell^{-1} \tilde{e_\ell} \bmod q$. Since the $\tilde{e_\ell}$s are uniformly distributed, the inputs to g are uniformly distributed as well. Applying the so-called *hybrid* technique, which amounts to using n times the *almost-invertibility* of g, we bound the difference between the success probabilities of the two games by $n\delta_g$, and thus: $|\Pr[S_3] - \Pr[S_2]| \leq n\delta_g$.

Game_4: In this game, we simulate the signing oracle. For any query m_j to the signing oracle, one computes $e_j = H(m_j)$, and issues a random signature (r_j, s_j), using the simulation of Lemma 3. Recall that the simulation picks s_j at random and computes r_j as $f(u_j)$, where u_j is randomly drawn from S. By Lemma 3, this simulation is perfect. Observe that, while checking the signature, the adversary requests, at some later time, the encoding of $e_j {s_j}^{-1} + x r_j {s_j}^{-1} \bmod q$. We let ℓ the first index corresponding to such query, $F_\ell = b_\ell X + a_\ell$. We modify z_ℓ, replacing its earlier value by u_j and define $\tilde{e_\ell}$ as $e_j = H(m_j)$. Observe that we still have $f(z_\ell) = b_\ell a_\ell^{-1} \tilde{e_\ell} \bmod q$. This

game only differs from the previous one if polynomial $e_j {s_j}^{-1} + X r_j {s_j}^{-1}$ collides with a previous one. Due to the randomness of s_j, we can bound $|\Pr[S_4] - \Pr[S_3]| \leq n q_s/q$.

We note that the final simulation runs in time $t' \leq t + n \times (5\tau_g \ln n + \tau_H)$ and we finally upper-bound $\Pr[S_4]$. We observe that, while checking the signature, the final request of the adversary, with index $n+2$, is the encoding of $es^{-1} + xrs^{-1} \bmod q$, where $e = H(m)$. We let ℓ be the first occurrence of F_{n+2}. If the signature is valid, the following equalities hold:

$$es^{-1} = a_\ell \bmod q, \quad rs^{-1} = b_\ell \bmod q, \quad f(z_\ell) = b_\ell a_\ell^{-1} \tilde{e_\ell} \bmod q \text{ and } r = f(z_\ell).$$

From these equalities, it easily follows that $r = f(z_\ell) = re^{-1}\tilde{e_\ell} \bmod q$, which in turn implies $e = \tilde{e_\ell} \bmod q$. We distinguish two cases:

- If z_ℓ has been created according to the rule of Game_3, then a pre-image m of some randomly chosen element $\tilde{e_\ell}$ among the n possible ones has been found.
- If z_ℓ has been created according to the rule of Game_4, then $\tilde{e_\ell} = e_j$ is the image under H of a message m_j queried from the signing oracle. Furthermore, we have: $e_j s_j^{-1} = a_\ell \bmod q$ and $r_j s_j^{-1} = b_\ell \bmod q$. Comparing to the above equalities, we get that $s = s_j \bmod q$ and $r = r_j \bmod q$. Note that m_j cannot be equal to m, since otherwise the output forged signature would coincide with an earlier signature (r_j, s_j) of the same message m. Thus, a collision has been found for H', where $H'(m) \stackrel{\mathsf{def}}{=} H(m) \bmod q$.

The probability that an algorithm running in time t' finds a preimage under H' of an element among n is at most $n\varepsilon_H$. From this, we obtain that: $\Pr[S_4] \leq n\varepsilon_H + \gamma_H$. Summing up inequalities, we get the announced result. □