Message Recovery Signature Schemes from Sigma-protocols

Masayuki Abe†, *Tatsuaki Okamoto, and Koutarou Suzuki*

Abstract

This paper presents a framework for constructing message recovery signature schemes from sigma-protocols. The heart of our construction is a redundancy function that adds some redundancy to a message that only a legitimately signed and recovered message can have. We characterize redundancy functions that make the resulting message recovery signature scheme proven secure. Our framework includes known schemes when the building blocks are given concrete implementations, so it presents insightful explanation into their structure.

1. Introduction

Digital signatures have been a central research subject in cryptography and are widely used these days. Among the many theoretical issues to pursue, efficiency is of great concern in practice. It is typically measured by the size of the public key and the signatures and by the computational cost for signature generation and verification. This paper focuses on efficient digital signature schemes that yield short signatures.

A digital signature scheme yields either a digital signature *with an appendix* or *with message recovery*. In the former case, a signed message looks like (σ, *msg*), where σ is the signature and *msg* is the message. Given (σ, *msg*), the verifier can determine whether σ is a correct signature that guarantees the authenticity and integrity of message *msg*. Naturally, the length of a signed message is the sum of the length of the message and the signature. In the latter case, a signed message is like (σ, n), where n is *part of* the message. The remaining part of the message is contained in signature σ, and anyone can recover the entire message. Because of this structure, one can expect a signed message to be shorter than a message in the case of a signature with appendix.

† NTT Information Sharing Platform Laboratories
Musashino-shi, 180-8585 Japan
Email: abe.masayuki@lab.ntt.co.jp

Typically, a message recovery signature proceeds as follows. The signer adds some redundancy to the message to sign it. This produces a signed message (σ, n). To verify the signed message, the verifier first recovers the entire message *msg* and then checks whether the recovered message has the correct redundancy. All that is required to recover the message is the public key, so anyone can do it. Furthermore, every random string (σ, n) will yield a string that looks like a redundant message (but is actually a random meaningless one). Accordingly, it is essential to choose redundancy that cannot be easily satisfied by the message recovered from forged (σ, n).

However, there are many message recovery signature schemes in the literature that do not specify or even characterize the necessary properties of the redundancy that makes the signature scheme proven secure. For RSA (Rivest, Shamir, and Adleman) signatures, there are some redundancy functions (also known as the padding method) such as PSS-R [1] that eventually achieves security against adaptive chosen message attacks [2] in the random oracle model [3]. In [4]–[6], Nyberg and Rouppel presented message recovery signature schemes whose security is based on the discrete-log problem, but no specific redundancy function was given, so its security is unclear. It was followed by some similar schemes, e.g., [7], also without specific redundancy functions. In [8], Abe and Okamoto presented a message recovery version of the Schnorr signature scheme [9] with a specific

redundancy using random oracles but with a very complicated security analysis that results in a high cost for reduction to the discrete logarithm problem. In [10], Naccache and Stern constructed a message recovery signature scheme from DSA [11] with characterization of sufficient redundancy. Pintsov and Vanstone presented a message recovery version of the Schnorr signature scheme whose security is proven in the ideal cipher model, which is known as the ECPV scheme [12], [13]. Many of these schemes are included in international standards such as [14].

The Fiat-Shamir transform [15] is a well-known methodology for obtaining a secure digital signature scheme. It converts a *sigma-protocol* [16], [17] into a secure signature scheme in the random oracle model. A sigma-protocol is a two-party protocol where a prover interacts with a verifier to convince him that the prover knows a secret, which is often understood as a secret key corresponding to a public key, without leaking anything about the secret. More precisely, it is a public-coin three-round honest verifier zero-knowledge proof system that has special soundness and special honest verifier zero-knowledge. Such protocols are often used for interactive authentication. The Fiat-Shamir transform can be used to eliminate the interaction and enable the prover alone to create a transcript of the execution of the sigma-protocol so that it can act as evidence of his knowledge about the secret, like the interactive version. When an arbitrary message is bound to the transcript, one can see that only the person who knows the secret can create the transcript with the message. Thus, it works as a digital signature with appendix. However, it is not known whether such a technique is also useful for designing digital signature schemes with message recovery.

In this paper, we present a framework for constructing message recovery signature schemes from sigma-protocols. Our results show how to securely *convolute* part of a document into a message used in the sigma-protocol by using a redundancy function and eventually turn the sigma-protocol into a message recovery signature scheme that is adaptively chosen message secure in the random oracle model.

Importantly, we characterize the redundancy function that is sufficient to make the resulting message recovery signature scheme proven secure in the random oracle model. We then show two instantiations of the redundancy function that conforms to our requirements based on the random oracle model and the ideal cipher model. At 80-bit security with typical parameter settings, both instantiations shorten the signed message by 80 bits. Our framework yields known schemes, ECPV in particular, when the building blocks are given concrete implementations; hence, it presents insightful explanation into their structure.

2. Preliminaries

2.1 Notation

We consider uniform probabilistic algorithms (i.e., Turing machines) that take as input (the unary encoding of) a security parameter $\lambda \in \mathbb{N}$ and possibly other inputs and run in deterministic polynomial time in λ. We thus always implicitly require the size of the input to be bounded by some polynomial in λ. Adversarial behavior is modeled by *non-uniform* polynomial-time probabilistic algorithms, i.e., by algorithms that together with the security parameter λ also get a polynomial-sized auxiliary input aux_λ. To simplify the notation, we usually let the dependency on λ (and on aux_λ) remain implicit. By $\mathcal{A}(x)$, we denote the output distribution of a probabilistic algorithm $\mathcal{A}$ with input x and uniformly chosen randomness. By $y \leftarrow \mathcal{A}(x)$, we mean that algorithm $\mathcal{A}$ is executed on input x and the output is assigned to y. We may also denote it as $y \leftarrow \mathcal{A}(x; r)$ when the randomness r is to be explicitly noted. For any algorithm $\mathcal{A}$, the output domain of $\mathcal{A}$ is denoted by $Dom(\mathcal{A})$. Similarly, for any finite set S, we use the notation $y \leftarrow S$ to denote that y is sampled uniformly from S, and $y \leftarrow x$ means that the value x is assigned to y. By $BitLen(S)$, we denote the number of bits used to present the largest element in S.

$|X|$ is used in different ways. If X is a set, $|X|$ denotes the cardinality of X. If X is a distribution or an algorithm, $|X|$ denotes the number of elements in X with nonzero probability. If X is a variable whose actual value is taken from some range, $|X|$ denotes the number of bits needed to present the maximum value in the range. (Hence, $|X|$ does not necessarily equal $[\log_2 X]$ for the actual value assigned to X.) Finally, if X is a string, $|X|$ is the number of bits needed to present the string.

We use $X := Y$ to define a new variable X with Y. For any two strings a and b, $a||b$ denotes a string that concatenates a and b in order. By $a||b \leftarrow c$, we denote that string c is separated into two parts and assigned to a and b. The manner of separation depends on the context but may not be explicitly given when it is clear from the context.

$P[y = \mathcal{A}(x)]$ denotes the probability (taken over a uniformly distributed random tape) that $\mathcal{A}$ outputs y on input x, and we write $P[x \leftarrow \mathcal{B}: \mathcal{A}(x) = y]$ for the

(average) probability that $\mathcal{A}$ outputs y on input x when x is output by $\mathcal{B}$: $P[x \leftarrow \mathcal{B}: \mathcal{A}(x) = y] = \sum_x P[y = \mathcal{A}(x)]P[x = \mathcal{B}]$. We also use natural self-explanatory extensions of this notation.

An *oracle* algorithm $\mathcal{A}$ is an algorithm in the above sense connected to an oracle in that it can write on its own tape an input for the oracle and tell the oracle to execute it. Then, in a single step, the oracle processes this input in the prescribed way and writes its output on the tape. We write $\mathcal{A}^{\mathcal{O}}$ when we consider $\mathcal{A}$ to be connected to the particular oracle $\mathcal{O}$.

As is common practice, a value $v(\lambda) \in R$, which depends on the security parameter λ, is treated as *negligible*, denoted by $v(\lambda) \leq negl(\lambda)$ or $v \leq negl$, if $\forall c > 0\ \exists \lambda' \in \mathbb{N}\ \forall \lambda \geq \lambda'$: $v(\lambda) < 1/\lambda^c$.

2.2 Sigma-protocol

Let $L \subset \{0, 1\}^*$ be a language and R_L be a binary relation associated with L. Let $W_L(x)$ for $x \in L$ denote a set of witnesses for x, i.e., $W_L(x) = \{w | R_L(x, w) = 1\}$. Let $\mathcal{I}$ be an instance generator for L that takes security parameter λ and chooses an instance $x \in L$ and a witness $w \in W_L(x)$ with restriction $|x| = \lambda$. A sigma-protocol for language L is an honest verifier zero-knowledge proof system that consists of (probabilistic) polynomial-time algorithms (A, C, Z, V, E, M). These algorithms work as follows.

For $(x, w) \leftarrow \mathcal{I}(\lambda)$, the verifier is given x and the prover is given (x, w). The prover first sends the first message $a \leftarrow A(x; t)$ to the verifier and the verifier challenges the prover by returning $c \leftarrow C(x)$. The prover answers the challenge by sending $z \leftarrow Z(w, t, c)$ to the verifier. The verifier accepts if $a = V(x, c, z)$. (More generally, V can be an algorithm that takes (a, x, c, z) as input and returns 1 or 0 to report acceptance and rejection. Note, however, that it is essential for our purpose that a can be computed from (x, c, z) in polynomial time.) We assume *perfect correctness*, where the verifier accepts with probability 1 when both the prover and verifier are honest.

A sigma-protocol features *special soundness*. That is, there exists a deterministic polynomial-time algorithm E, called an extractor, that takes two acceptable transcripts with the same first message and different challenges, say (x, a, c, z) and (x, a, c', z') such that $x \in L$, $c \neq c'$, $V(x, c, z) = V(x, c', z') = a$ as input and outputs a witness w for x with probability 1 in polynomial time.

A sigma-protocol is *honest verifier zero-knowledge*. Namely, we assume that there exists a polynomial-time algorithm called a simulator, denoted by M, that takes $x \in L$ and $c \leftarrow C(x)$ as input and outputs (a, z) such that the distribution of simulated (a, c, z) is identical to the distribution of the real transcript generated by the honest prover and verifier.

A note on relaxation. We can allow a small error probability in the correctness but the resulting signature scheme inherits the error probability in its correctness. Similarly, E can be a probabilistic algorithm with negligible error probability. Also, the quality of zero-knowledge simulator M can be relaxed to statistical or computational rather than perfect. These relaxations affect the unforgeability of the resulting signature scheme.

2.3 Security model of message recovery signature scheme

To define the security precisely, we first must present a syntactical definition of message recovery signature schemes. The one shown below is a very standard definition, which can be seamlessly turned into the definition of ordinary signature schemes with appendix by letting the non-recoverable message n be the same as the entire message msg.

Definition 1. (Message recovery signature scheme) A message recovery signature scheme consists of three probabilistic polynomial-time algorithms $(\mathcal{G}, \mathcal{S}, \mathcal{V})$.

- $\mathcal{G}$ is a key generation algorithm that takes security parameter λ and outputs a public-key pk and a private-key sk. Associated with pk is a recoverable message space $\{0, 1\}^{\ell_{\mathrm{rec}}}$.
- $\mathcal{S}$ is a signature-issuing algorithm that takes a private-key sk and a message msg and outputs a signature σ and the non-recoverable part of the message n.
- $\mathcal{V}$ is a verification protocol that takes a public key pk and a signed message (σ, n) and outputs (accept, msg) or reject. It is required that (accept, msg) $= \mathcal{V}(pk, \mathcal{S}(sk, msg))$ for any pk and sk generated by $\mathcal{G}$ and for any msg.

A message recovery signature scheme is secure if it withstands an adversary that asks a legitimate signer to sign arbitrary messages and then attempts to forge a valid signed message never signed by the signer. Formal definition of such a chosen message adversary follows.

Definition 2. (Chosen message adversary) A $(\tau_{\mathrm{sig}}, \epsilon_{\mathrm{sig}}, q_s, q_h)$ chosen message adversary A_{sig} against the above message recovery signature scheme is an oracle Turing machine that launches a chosen message attack that consists of the following steps.

1. $(pk, sk) \leftarrow \mathcal{G}(\lambda)$
2. $(\tilde{\sigma}, \tilde{n}) \leftarrow A_{sig}^{\mathcal{S},\mathcal{H}}(pk)$

$\mathcal{S}$ is the signing oracle such that, for every signing query for a message, say msg_i, it returns a legitimately generated signed message (σ_i, n_i) by using private key sk. $\mathcal{H}$ is the random oracle that corresponds to hash function H in the signature scheme. Let T denote the signed messages (σ_i, n_i) observed by $\mathcal{S}$. The final output of A_{sig} must satisfy $(\tilde{\sigma}, \tilde{n}) \notin T$. A_{sig} makes at most q_s queries to $\mathcal{S}$ and at most q_h queries to $\mathcal{H}$ and then stops within running time τ_{sig}. Then, $\mathcal{V}(pk, \tilde{\sigma}, \tilde{n})$ outputs (accept, msg) for some msg with probability at least ϵ_{sig}. The probability is taken over all the random coins used in the chosen message attack.

3. Our construction

3.1 Redundancy function

The heart of our construction is the redundancy function, say Δ: $Dom(A) \times \{0, 1\}^{\ell_{\mathrm{rec}}} \to D$, that takes the first message a of the underlying sigma-protocol and a recoverable part of the message m and outputs a redundant message r in some specific domain D. More precisely, we assume a family of functions $\Delta_\lambda = \{\Delta\}$ and its ensemble $\{\Delta_\lambda\}_{\lambda \in \mathbb{N}}$ that satisfy the following properties.

1. (Invertibility) For every Δ, there exists an inverse function Δ^{-1} that, given (a, r), outputs m. It is required that $m = \Delta^{-1}(a, \Delta(a, m))$.
2. (Compactness) $BitLen(Dom(A)) + \ell_{\mathrm{rec}} \geq BitLen(D)$.
3. (Randomness) For every Δ and every m, the distribution of $\Delta(a, m)$ for uniformly chosen a is uniform.
4. (Collision resistance) Given $\Delta \leftarrow \Delta_\lambda$, any probabilistic algorithm running within time τ finds (a, m) and (a', m') such that $\Delta(a, m) = \Delta(a', m')$ and $(a, m) \neq (a', m')$ only with probability, say $\epsilon_\Delta^{\mathrm{coll}}(\tau)$, which is negligible in λ if τ is polynomial in λ.

Invertibility is needed for our construction to work. Compactness is needed for the message recovery property to be meaningful. Without this property, the signed message of the resulting message recovery signature scheme would be longer than the one from the signature scheme with appendix. Randomness and collision resistance are needed for unforgeability of the resulting signature scheme.

One can relax the randomness property to be computationally indistinguishable from uniform. This will affect the unforgeability of the resulting signature scheme, as we mention later.

As we shall see in Section 4. $\epsilon_\Delta^{\mathrm{coll}}(\tau)$ may take some other parameters such as the maximum number of oracle queries, depending on its construction.

3.2 Our scheme

Given a sigma-protocol (A, C, Z, V, E, M) for $\mathcal{I}$ and a hash function H, we construct a message recovery signature scheme $(\mathcal{G}, \mathcal{S}, \mathcal{V})$ as follows.

- (Key generation: $\mathcal{G}$) Given λ, run $(x, w) \leftarrow \mathcal{I}(\lambda)$ and output x as a public key and w as a private key. Security parameter λ also determines the length of the recoverable part ℓ_{rec} and the redundancy function Δ.
- (Signature generation: $\mathcal{S}$) Given private key w and message msg, first chop msg as $msg = m||n$ so that $m \in \{0, 1\}^{\ell_{\mathrm{rec}}}$. Then, compute $a \leftarrow A(x; t)$, $r \leftarrow \Delta(a, m)$, $c \leftarrow H(r, n)$, and $z \leftarrow Z(w, t, c)$. Then, output (r, z, n) as a signed message.
- (Signature verification: $\mathcal{V}$) Given x and (r, z, n), compute $c \leftarrow H(r, n)$, $a \leftarrow V(x, c, z)$, and $m \leftarrow \Delta^{-1}(a, r)$. If $r \leftarrow \Delta(a, m)$, output (accept, $m||n$). Otherwise, output reject.

Implementation note. Depending on the specific structure of function Δ, the signature verification can be more efficient than recomputing Δ. See also Subsections 4.1 and 4.2.

3.3 Proof of security

Theorem 1. (Security against chosen message attacks) If there exists $(\tau_{\mathrm{sig}}, \epsilon_{\mathrm{sig}}, q_s, q_h)$-*adversary* A_{sig} for the message recovery signature scheme based on language L and an instance generator $\mathcal{I}$, then there exists a (τ_w, ϵ_w) witness extraction adversary $A_{\mathcal{I}}$, where $\tau_w \leq 8\,\tau_{\mathrm{sim}}/(\epsilon_{\mathrm{sig}} - 4\,\frac{q_s q_h}{|D|})$ and $\epsilon_w \geq \frac{9}{100} \cdot \frac{1}{q_h}\,(1 - \frac{1}{|C|})\,(1 - \epsilon_{\Delta,\,\mathrm{sim}}^{\mathrm{coll}})$.

Here, τ_{sim} is $\tau_{\mathrm{sig}} + q_s T_{\mathrm{sig}}$, where T_{sig} is the sum of the running times of C, M, and Δ, and $\epsilon_{\Delta,\,\mathrm{sim}}^{\mathrm{coll}}$ is $(\frac{\epsilon_{\mathrm{sig}}}{2} - \frac{q_s q_h}{|D|})^{-1} \epsilon_\Delta^{\mathrm{coll}}(2\tau_{\mathrm{sim}})$.

Proof. This is proved by constructing $A_{\mathcal{I}}$ from A_{sig} using a forking lemma in a standard manner. Let $x \in L$ be an instance generated by $\mathcal{I}(\lambda)$. Given x as input, $A_{\mathcal{I}}$ works as follows.

1. Invoke A_{sig} with x as a public key and a uniformly chosen random tape. If A_{sig} terminates with a valid signed message $(\tilde{r}, \tilde{z}, \tilde{n})$, proceed to the next step. Otherwise, repeat this step with the same x and a fresh random tape up to t_1 times. Abort if never successful. Oracle queries are handled as follows.

- (Query msg_i to $\mathcal{S}$.) Execute $c_i \leftarrow C(x)$ and $(a_i, z_i) \leftarrow M(x, c_i)$. Then, chop msg_i into $m_i||n_i$, compute $r_i \leftarrow \Delta(a_i, m_i)$, and define H so that $c_i = H(r_i, n_i)$.

If H has already been defined for such an input, abort. Otherwise, return (r_i, z_i, n_i) as a signed message.

– (Query (r_j, n_j) to $\mathcal{H}$.) If the input is fresh, select c_j uniformly, define H as $c_j = H(r_j, n_j)$, and return c_j. Otherwise, return the preliminarily defined value.

2. Let i^* denote the index that $\mathcal{H}$ is asked $(\tilde{r}, \tilde{n})$. That is, $r_{i^*} = \tilde{r}$, $n_{i^*} = \tilde{n}$, and $c_{i^*} = H(\tilde{r}, \tilde{n})$. Let $\tilde{a}$ denote $\tilde{a} \leftarrow V(x, c_{i^*}, \tilde{z})$. Move to the next step.
3. Invoke A_{sig} with x and the random tape used in the successful run of the first step. The simulation for oracle $\mathcal{S}$ is unchanged. Oracle $\mathcal{H}$ is simulated identically up to the point that $(\tilde{r}, \tilde{n})$ is asked. From that moment, $\mathcal{H}$ is simulated with new randomness. If A_{sig} terminates with a valid signed message $(\tilde{r}, \tilde{z}', \tilde{n})$, then proceed to the next step. Otherwise, repeat this step up to t_2 times. Abort if never successful.
4. Let $c'_{i^*} = H(\tilde{r}, \tilde{n})$, which was observed in the second step, and let $\tilde{a}'$ be $\tilde{a}' \leftarrow V(x, c'_{i^*}, \tilde{z}')$. If $c_{i^*} = c'_{i^*}$ or $\tilde{a} \neq \tilde{a}'$, abort. Otherwise, compute $w \leftarrow E(x, \tilde{a}, c_{i^*}, \tilde{z}, c'_{i^*}, \tilde{z}')$ and output w.

Now we estimate the running time and the probability of success of $A_{\mathcal{I}}$. First, the success probability is estimated as follows.

– First, we estimate the error probability of the oracle simulation. Observe that the simulation of $\mathcal{S}$ fails only if (r_i, n_i) has already been defined for H. Since the distribution of n_i is arbitrarily determined by the adversary, we consider only the sufficient condition that r_i has appeared so far.

Suppose that r_i is uniformly chosen from D. Then, the probability that r_i has appeared among the inputs to H is at most $q_h/|D|$.

Next consider r_i generated through Δ from uniformly chosen a_i. Since we assume that the distribution of the output of Δ is uniform, the above probability does not change at all. (This is where the relaxation of randomness on Δ would have an effect. If the randomness is not perfect but just indistinguishable with some advantage, say $\epsilon_{\Delta}^{\mathrm{dist}}(\tau)$ for a polynomial-time distinguisher whose running time is τ, then the error probability of the simulation increases by $\epsilon_{\Delta}^{\mathrm{dist}}(\tau_{\mathrm{sim}})$.)

Then, consider a_i generated by M. As in the previous step, since the distribution of such a_i is perfectly the same as uniform, the error probability does not change at all, either. (Again, this is where the relaxation on the zero-knowledge property of the underlying sigma-protocol matters. If the zero-knowledge simulation is only computational with some advantage, say at most $\epsilon_{\mathrm{zk}}(\tau)$ for a distinguisher whose running time is τ, then the error probability increases by $\epsilon_{\mathrm{zk}}(\tau_{\mathrm{sim}})$.)

Therefore, the probability ϵ_{serr} that the simulation fails while handling q_s queries is bound as

$$\epsilon_{\mathrm{serr}} \leq 1 - (1 - \frac{q_h}{|D|})^{q_s} \leq \frac{q_s q_h}{|D|}. \quad (1)$$

The simulation of random oracle $\mathcal{H}$ is obviously perfect unless the simulation of $\mathcal{S}$ fails.

– The success probability of A_{sig} for fixed x is at least $\epsilon_{\mathrm{sig}}/2$ with probability 1/2 (over the choice of x) due to the heavy-row lemma of [15], [18]. With the simulated signing oracle, the success probability reduces to $\epsilon_{\mathrm{sig}}/2 - \epsilon_{\mathrm{serr}}$. Accordingly, every run of A_{sig} in the first step is successful with probability at least $\mu_1 := (\epsilon_{\mathrm{sig}}/2 - \epsilon_{\mathrm{serr}})$. By repeating the attempt up to $t_1 := 1/\mu_1$ times, we get at least one valid $(\tilde{r}, \tilde{z}, \tilde{n})$ at the end of the first step with probability greater than $\epsilon_1 \geq 1 - (1 - \mu_1)^{1/\mu_1} \geq 1 - e^{-1} \geq 3/5$. Let Θ^* denote the randomness used for simulating the random oracle for the i^*-th and all subsequent queries in the successful run. Let Θ denote all other randomness also used in the successful run.
– Over the choice of Θ^*, the success probability of A_{sig} for fixed Θ and x is at least $\epsilon_{\mathrm{sig}}/4$ with probability 1/2 (over the choice of Θ) due to the heavy-row lemma again. If we apply the same argument as in the first step, each attempt in the third step is successful with probability at least $\mu_2 := \epsilon_{\mathrm{sig}}/4 - \epsilon_{\mathrm{serr}}$. If we repeat this attempt up to $t_2 := 1/\mu_2$ times, another valid signed message $(\tilde{r}', \tilde{z}', \tilde{n}')$ is obtained with probability greater than $\epsilon_2 \geq 1 - (1 - \mu_2)^{1/\mu_2} \geq 1 - e^{-1} \geq 3/5$. Furthermore, the outcome corresponds to the i^*-th query to H, i.e., $\tilde{r}' = \tilde{r}$ and $\tilde{n}' = \tilde{n}$, with probability at least $1/q_h$.
– In the fourth step, $c_{i^*} = c'_{i^*}$ happens only with probability $1/|C|$. Observe that, if $\tilde{a} \neq \tilde{a}'$, then we have $\tilde{r} = \Delta(\tilde{a}, \Delta^{-1}(\tilde{a}, \tilde{r})) = \Delta(\tilde{a}', \Delta^{-1}(\tilde{a}', \tilde{r}))$ which is a collision. Let $\epsilon_{\Delta,\mathrm{sim}}^{\mathrm{coll}}$ denote the probability of such an event occurring. Since the running time of $A_{\mathcal{I}}$ is less than $\tau_w = (t_1 + t_2)\,\tau_{\mathrm{sim}}$, $\epsilon_{\Delta,\mathrm{sim}}^{\mathrm{coll}}$ can be upper bounded by $\epsilon_{\Delta}^{\mathrm{coll}}(\tau_w)$. It is, however, an overestimation. More precise estimation of the probability is at most

$$\epsilon_{\Delta,\mathrm{sim}}^{\mathrm{coll}} = t_1 \cdot \epsilon_{\Delta}^{\mathrm{coll}}(2\tau_{\mathrm{sim}}) \quad (2)$$

because we have at most t_1 trials for an independent choice of Δ, and the particular Δ chosen in the successful run in the first step is executed again in the third step of the simulation and the total running time with this particular Δ is $2\tau_{\mathrm{sim}}$.

By summing up the above assessment, we get the total success probability of $A_{\mathcal{I}}$ as

$$\epsilon_w \geq \frac{1}{2} \cdot \frac{3}{5} \cdot \frac{1}{2} \cdot \frac{3}{5} \cdot \frac{1}{q_h} (1 - \frac{1}{|C|}) (1 - \epsilon^{\text{coll}}_{\Delta, \text{sim}}) \quad (3)$$

$$\geq \frac{9}{100} \cdot \frac{1}{q_h} (1 - \frac{1}{|C|}) (1 - \epsilon^{\text{coll}}_{\Delta, \text{sim}}). \quad (4)$$

Assuming that the time for generating randomness and table searching used for simulating H is free, the running time is

$$\tau_w \leq (t_1 + t_2) \cdot \tau_{\text{sim}} \quad (5)$$

$$\leq (1/(\epsilon_{\text{sig}}/2 - \epsilon_{\text{serr}}) + 1/(\epsilon_{\text{sig}}/4 - \epsilon_{\text{serr}})) \cdot \tau_{\text{sim}} \quad (6)$$

$$\leq 8\, \tau_{\text{sim}} /(\epsilon_{\text{sig}} - 4\, \frac{q_s q_h}{|D|}) \quad (7)$$

4. Construction of Δ

4.1 In the random oracle model

We construct Δ and Δ^{-1} as follows.

$$\Delta(a, m) = h_1||h_2, \text{ where } h_1 = H_1(a, m) \text{ and } h_2 = m \oplus H_2(a, h_1) \quad (8)$$

$$\Delta^{-1}(a, h_1||h_2) = h_2 \oplus H_2(a, h_1) \quad (9)$$

H_1: $\{0, 1\}^* \to \{0, 1\}^{\ell_1}$ and H_2: $\{0, 1\}^* \to \{0, 1\}^{\ell_{\text{rec}}}$ are hash functions modeled as independent random oracles.

To make the message recovery property meaningful, $\ell_1 < |a|$ should hold. Note that, as we shall show in Lemma 2, setting ℓ_{rec} too short would cause a security problem, i.e., it would result in a high probability of collision. Recommendable settings of ℓ_1 and ℓ_{rec} are $\ell_1 = \ell_{\text{rec}} = |a|/2$ to balance security and the benefit of message recovery. If one would like the length of the recoverable part to vary depending on the given message, the message must be polynomially long or some kind of padding must be applied so that the total length becomes polynomial in λ.

With the above instantiation, the verification procedure specified in Subsection 3.2 can be slightly more efficient. The modified signature verification procedure is as follows.

– (Signature verification: $\mathcal{V}$) Given x and (r, z, n), compute $c \leftarrow H(r, n)$, $a \leftarrow V(x, c, z)$, and $m \leftarrow \Delta^{-1}(a, r)$. Then, let $h_1||h_2 \leftarrow r$. If $h_1 = H_1(a, m)$, output (accept, $m||n$). Otherwise, output reject.

Namely, it replaces the testing $r = \Delta(a, m)$ with $h_1 = H_1(a, m)$ and saves one computation of H_2.

The above Δ satisfies the required properties listed in Subsection 3.1. Specifically, the following lemmas hold. The first three (invertibility, compactness, and randomness) are trivial and the last one (collision resistance) is followed by a detailed proof.

Lemma 1. (Invertibility, compactness, and randomness) The above Δ is invertible. In particular, the above Δ^{-1} is the inverse function. It is compact when ℓ_1 is set so that $BitLen(Dom(A)) \geq \ell_1$. The output distribution of Δ is uniform over $\{0, 1\}^{\ell_1 + \ell_{\text{rec}}}$ if H_1 and H_2 are independent random oracles.

Lemma 2. (Collision resistance) If ℓ_1 and ℓ_{rec} are polynomial in λ, the above Δ is collision resistant against polynomial-time adversaries that make a polynomial number of queries to H_1 and H_2. In particular, for any adversary asking q_1 and q_2 queries to H_1 and H_2, respectively, the probability of collision is upper bounded by $\epsilon^{\text{coll}}_{\Delta}(q_1, q_2) \leq \frac{q_1}{2^{\ell_1 + \ell_{\text{rec}}}} + \frac{q_1}{2^{\ell_1}} + \frac{q_1 q_2}{2^{\ell_1 + \ell_{\text{rec}}}} + \frac{q_1}{2^{\ell_1 + \ell_{\text{rec}}}} + \frac{q_1 q_2}{2^{2\ell_1}}$.

Proof. (of Lemma 2) We refine the notation for the input and output of H_1 and H_2 as follows. Let $h_i = H_1(R_i, M_i)$ and $c_j = H_2(D_j, e_j)$ denote the i-th and j-th input/output of H_1 and H_2, respectively. We first define the notation and some terminology. Let L_1 denote the set of queries and answers for H_1, i.e., $L_1 = \{(R_i, M_i, h_i) | i = 1, ..., q_1\}$. Define $L_2 = \{(D_j, e_j, c_j) | j = 1, ..., q_2\}$ as well. Without loss of generality, we assume that every query is fresh, so every entry in L_1 and L_2 is unique. By (i, j), we denote a set that consists of the i-th entry of L_1 and the j-th one of L_2, i.e., $(i, j) = (R_i, M_i, h_i, D_j, e_j, c_j)$. We say that (i, j) is fully consistent if $(R_i, h_i) = (D_j, e_j)$. Note that if (i, j) is fully consistent, it forms a correct computation of Δ. By FullCon $(i, j) = \mathsf{true}$ or false, we mean that (i, j) is or is not fully consistent, respectively. We also say that (i, j) is *semi-consistent* if $R_i = D_j$. A semi-consistent (i, j) may also be fully consistent but this is not necessary. By SemiCon$(i, j) = \mathsf{true}$ or false, we mean that (i, j) is or is not semi-consistent, respectively. Let $\tau(H_b, i)$ for $b \in 1, 2$ denote the time at which the i-th query is made to H_b. (It can be understood as the number of steps before the i-th query is made to oracle H_1). Since no more than one query can be made at one step, $\tau(H_b, i)$ is strictly larger or smaller than $\tau(H_{b'}, i')$ if $i \neq i'$. We say that (i, j) is *regular* if $\tau(H_1, i) < \tau(H_2, j)$; otherwise, it is *irregular*. The intuition behind this terminology is that anyone who computes Δ correctly should ask H_1 first and then H_2 in that order. Therefore, a normal computation of Δ yields regular and fully consistent (i, j). By Regular$(i, j) =$

true or false, we mean that (i, j) is regular or irregular, respectively. We define function $X(i, j)$ as $X(i, j) = c_j \oplus M_i$.

With this terminology, we can say that a collision $\Delta(R_i, M_i) = \Delta(R_u, M_u)$ happens if and only if there exist j and v such that

– $(i, j) \neq (u, v)$,
– FullCon(i, j) = FullCon(u, v) = true, and
– $h_i||X(i, j) = h_u||X(u, v)$.

Accordingly, we say that (i, j) and (u, v) collide if and only if the above conditions are satisfied. By Collision(i, j, u, v) = true or false, we mean (i, j) and (u, v) do or do not collide, respectively.

We classify entries in L_2 by the value of e_j. Namely, for every different value of e_j in L_2, we consider a set of indexes $\{v|e_v = e_j\}$ and label such sets as Class$_1$, Class$_2$, ... In this way, every entry in L_2 is associated with one class. Obviously, there are at most q_2 classes. Let SameClass(j, v) be true if and only if j and v are in the same class, i.e., $e_j = e_v$.

The above definitions lead simply to some facts.

Fact 1. If SameClass(j, v) = false, then Collision(i, j, u, v) = false.

If two consistent (i, j) and (u, v) are in different classes, then $h_i = e_j \neq e_v = h_u$ holds and they cannot collide. Moreover, if either of them is inconsistent, it cannot have a collision, either. Therefore, we consider the possibility of collision only within a class.

Fact 2. $|\{(i, j)|\text{FullCon}(i, j) = \text{true}\}| \leq q_1$.

Namely, there are at most q_1 pairs of fully consistent queries. Suppose that (i, j) is fully consistent and (i, j') is also fully consistent for some j'. Then, $e_j = h_i = e_{j'}$ and $D_j = R_i = D_{j'}$ holds, and it means that the j-th and j'-th queries to H_2 are identical, whereas we assumed that L_2 contains no duplicate entries.

Fact 3. $|\{(i, j)|\text{SemiCon}(i, j) = \text{true}, \wedge j \in \text{Class}_k\}| \leq q_1$ for every Class$_k$.

Namely, there are at most q_1 semi-consistent pairs of queries with regard to each class of L_2. Suppose that (i, j) and (i, j') are both semi-consistent and j and j' are in the same class. We then have $D_j = R_i = D_{j'}$ and $e_j = e_{j'}$. Hence, the j-th and j'-th entries in L_2 are identical.

Suppose that (i, j) and (u, v) are both semi-consistent and they are in the same class. These queries can collide. (On the other hand, if either of them is not semi-consistent or if they are in different classes, they cannot collide.) Regarding the regularity of (i, j) and (u, v), one of the following cases must be true.

Case 1. Both are regular.
Case 2. One is regular and the other is irregular.
Case 3. Both are irregular.

A pair of regular queries, say (i, j), is either fully consistent (the case where (R_i, h_i) is the input (D_i, e_i) to H_2) or inconsistent (something else is given as input to H_2). These cases are independent of the choice of the randomness of H_1 and H_2. If (i, j) is regular but inconsistent, there is no chance of it causing a collision with another pair of queries. Hence, we only need to consider fully consistent queries in the regular case. Below, we consider the probability of collision in each of the above cases.

Case 1. Consider regular and fully consistent queries (i, j) and (u, v) in the same class. For a collision to occur, $h_i = h_u$ and $c_j \oplus M_i = c_v \oplus M_u$ are necessary. First of all, $h_i = h_u$ happens with probability at most $\frac{1}{2^{\ell_1}}$ because both h_i and h_u are uniformly chosen as a result of the true randomness of H_1 and H_2. Next, consider $c_j \oplus M_i = c_v \oplus M_u$. Since (i, j) and (u, v) are regular, $\tau(M_i) < \tau(c_j)$ and $\tau(M_u) < \tau(c_v)$ hold. Therefore, at least one of c_j and c_v is independent of M_i and M_u. Since both c_j and c_v are chosen uniformly, $c_j \oplus M_i = c_v \oplus M_u$ holds with probability at most $\frac{1}{2^{\ell_{\text{rec}}}}$. Thus, (i, j) and (u, v) collide with probability at most $\frac{1}{2^{\ell_1+\ell_{\text{rec}}}}$. Let FullCon(Class$_k$) denote the number of fully consistent queries in Class$_k$. By summing up the probability for all the classes and applying Fact 2, we get

$$\Pr[\text{Case 1}] \leq \sum_k \frac{\text{FullCon}(\text{Class}_k)^2}{2^{\ell_1+\ell_{\text{rec}}}} \leq \frac{{q_1}^2}{2^{\ell_1+\ell_{\text{rec}}}}. \quad (10)$$

Case 2. For every entry i in L_1, there exists at most one entry, say j, in each class of L_2 such that (i, j) is semi-consistent. (If there exists j' in the same class such that (i, j') is semi-consistent, then $j = j'$ because $R_i = D_j = D_{j'}$ and $e_j = e_{j'}$.) We consider the probability that the semi-consistent (i, j) in the class is irregular and collides with a regular and fully consistent (u, v) in the same class. First of all, (i, j) must be fully consistent to have a collision. Therefore, it must satisfy $h_i = e_j$ when h_i is randomly and independently chosen, which in turn happens with probability at most $\frac{1}{2^{\ell_1}}$. Second, $c_j \oplus M_i = c_v \oplus M_u$ must hold. Since (i, j) is irregular, however, M_i can be set after the values of c_j, c_v, and M_u have been seen, and this condition can be satisfied with probability 1 by setting M_i as $M_i = c_j \oplus c_v \oplus M_u$. Note that there is only one regular and fully consistent (u, v) in the class that causes $c_j \oplus M_i = c_v \oplus M_u$ with (i, j). Otherwise, there would exist another regular and fully consistent (u', v') in the same class

and it would cause $c_j \oplus M_i = c_{v'} \oplus M_{u'}$. Then, (u, v, u', v') would have a collision. This has already been treated in Case 1.

One question is whether or not such a manipulated M_i can also be a candidate for a collision in other classes. We claim that if (i, j) in Class_k has (u, v) in Class_k that satisfies $c_j \oplus M_i = c_v \oplus M_u$, then (i, j') in $\text{Class}_{k'}$ has (u', v') in $\text{Class}_{k'}$ that also causes $c_{j'} \oplus M_i = c_{v'} \oplus M_{u'}$ only with probability $\frac{1}{2^{\ell_{\text{rec}}}}$. (Naturally, we suppose (i, j') is irregular and semi-consistent and (u', v') is regular and fully consistent so they fall into Case 2.) This happens only if

$$c_j \oplus c_v \oplus M_u = c_{j'} \oplus c_{v'} \oplus M_{u'}. \tag{11}$$

Since (u, v) and (u', v') are regular, all $c_j\, c_v \oplus M_u\, c_{j'}$, and $c_{v'} \oplus M_{u'}$ are random and independent of each other. Hence, E_q.(11) is satisfied only with probability $\frac{1}{2^{\ell_{\text{rec}}}}$.

In summary, with regard to each query i in L_1, we have a collision probability of at most $\frac{1}{2^{\ell_1}} + \frac{(\text{no. of Classes})}{2^{\ell_1+\ell_{\text{rec}}}} \le \frac{1}{2^{\ell_1}} + \frac{q_2}{2^{\ell_1+\ell_{\text{rec}}}}$. Summing up the probabilities for all i in L_1, we get the upper bound of the probability for Case 2 as

$$\Pr[\text{Case 2}] \le \frac{q_1}{2^{\ell_1}} \le \frac{q_1 q_2}{2^{\ell_1+\ell_{\text{rec}}}}. \tag{12}$$

Case 3. We consider the probability that, for an irregular semi-consistent query (i, j) in a class, there exists another irregular semi-consistent query (u, v) in the same class that causes a collision.

First of all, (i, j) itself must be fully consistent. This is satisfied with probability at most $\frac{1}{2^{\ell_1}}$. In addition, (u, v) must be fully consistent and this happens with probability at most $\frac{1}{2^{\ell_1}}$.

Next, we consider the number of (u, v) in the class, say Class_k, that collide with (i, j) with nonzero probability. We first claim that if $j = v$, then (i, j) and (u, v) cannot collide. Suppose that $j = v$. Since both (i, j) and (u, v) are semi-consistent, $R_i = D_j = Dv = R_u$ holds.

Since (i, j) and (u, v) are distinct, $(R_i, M_i) \neq (R_u, M_u)$, so $M_i \neq M_u$. If a collision happens, $c_j \oplus M_i = c_v \oplus M_u$ must hold. But this is not possible because $cj = cv$ and $M_i \neq M_u$. We next claim that if (i, j) and (u, v) satisfy $c_j \oplus M_i = c_v \oplus M_u$, then no other (u', v) in the same class can cause $c_j \oplus M_i = c_v \oplus M_{u'}$. The reason is essentially the same as for the above claim. Since both (u, v) and (u', v) are semi-consistent and in the same class, $M_u \neq M_{u'}$ must be the case (otherwise, u and u' are the same query in L_1). Hence, $c_j \oplus M_i = c_v \oplus M_u = c_v \oplus M_{u'}$ cannot happen. From the above two claims, we can see that, for (i, j), and for every v in Class_k, there could exist at most one u that makes (i, j, u, v) have a collision. Namely, there are at most $|\text{Class}_k| - 1$ candidates of (u, v) that have nonzero probability of collision with (i, j).

In summary, for every irregular semi-consistent (i, j), the probability that there exists an irregular semi-consistent (u, v) that causes a collision is at most $\frac{1}{2^{\ell_1}} \cdot \frac{|\text{Class}_k| - 1}{2^{\ell_1}}$. From Fact 3, we have at most q_1 candidates of such (i, j) in a class. Thus, the probability of collision in each class is at most $\frac{q_1}{2^{\ell_1}} \cdot \frac{|\text{Class}_k| - 1}{2^{\ell_1}}$. Summing up the probabilities for all classes and applying $\sum_k |\text{Class}_k| \le q_2$, we get

$$\Pr[\text{Case 3}] \le \sum_k \frac{q_1}{2^{\ell_1}} \cdot \frac{|\text{Class}_k| - 1}{2^{\ell_1}} \le \frac{q_1 q_2}{2^{2\ell_1}}. \tag{13}$$

Summary. From the bounds shown as in the Eqs. (10), (12), and (13), we get

$$\begin{aligned} \epsilon_\Delta^{\text{coll}}(q_1, q_2) &= \Pr[\text{Case 1}] + \Pr[\text{Case 2}] + \\ \Pr[\text{Case 3}] &\le \frac{{q_1}^2}{2^{\ell_1+\ell_{\text{rec}}}} + \frac{q_1}{2^{\ell_1}} + \frac{q_1 q_2}{2^{\ell_1+\ell_{\text{rec}}}} + \frac{q_1 q_2}{2^{2\ell_1}} \end{aligned} \tag{14}$$

as stated.

4.2 In the ideal cipher model

Let $(\mathcal{SG}, \mathcal{SE}, \mathcal{SD})$ be a symmetric encryption such that: $\mathcal{SG}$ is a probabilistic algorithm that takes security parameter λ and outputs a secret key sk, $\mathcal{SE}$ is a (probabilistic) algorithm that encrypts input message msg and outputs a ciphertext ξ by using secret key sk, and $\mathcal{SD}$ is a decryption algorithm that takes ciphertext ξ and recovers plaintext msg by using secret key sk.

We model a symmetric encryption scheme as an ideal cipher in the following

- $\mathcal{SE}$ and $\mathcal{SD}$ are oracles.
- Given a query (sk, msg), oracle $\mathcal{SE}$ does the following. If $(sk, |msg|)$ is not recorded in Π, an initially empty list, select a new random permutation $\mathcal{RP}$: $\{0,1\}^{|msg|} \to \{0,1\}^{|msg|}$ and return $\mathcal{RP}(msg)$. Then, record $(sk, |msg|, \mathcal{RP})$ to Π. If $(sk, |msg|)$ is in Π, simply return $\mathcal{RP}(msg)$ using the corresponding $\mathcal{RP}$. Oracle $\mathcal{SD}$ does the reverse. That

is, given a query (sk, ξ), if $(sk, |\xi|)$ is not in Π, select a new random permutation $\mathcal{RP}$: $\{0,1\}^{|\xi|} \to \{0,1\}^{|\xi|}$, record $(sk, |\xi|, \mathcal{RP})$ in Π, and return $\mathcal{RP}^{-1}(\xi)$. If $(sk, |\xi|)$ is in Π, simply return $\mathcal{RP}^{-1}(\xi)$ using the corresponding $\mathcal{RP}$.

We now construct Δ and Δ^{-1} as follows. Let str be a sufficiently long fixed string.

$$\Delta(a, m) = \mathcal{SE}\,(sk, \mathsf{str}||m) \text{ where } sk \leftarrow \mathcal{SG}(\lambda; a). \quad (15)$$

$$\Delta^{-1}(a, \xi) = m \text{ where } var||m \leftarrow \mathcal{SD}(sk, \xi) \text{ and } sk \leftarrow \mathcal{SG}(\lambda; a). \quad (16)$$

Without loss of generality, we assume that $\mathcal{SG}$ takes randomness from the domain of a in the above construction. (Otherwise, a is preprocessed with an entropy smoothing method and then given to $\mathcal{SG}$.) The output of $\mathcal{SD}$ is separated into two parts so that the length of the first part equals the length of the prefixed string str and the remaining part is taken as a recovered message.

In Δ^{-1}, the recovered str is not used at all. It can be used to make the verification procedure slightly more efficient as follows.

– (Signature verification: $\mathcal{V}$) Given x and (r, z, n), compute $c \leftarrow H(r, n)$, $a \leftarrow V(x, c, z)$, and $m \leftarrow \Delta^{-1}(a, r)$. Then, let var be the string obtained while computing Δ^{-1}. If $var = \mathsf{str}$, output (accept, $m||n$). Otherwise, output reject.

Namely, it verifies whether or not the predetermined string str is recovered correctly.

We claim that the above construction conforms to our requirements. Again, the first three requirements are obviously satisfied. Collision resistance is stated with a proof below. Let ℓ_{str} be $BitLen(\mathsf{str})$.

Lemma 3. (Invertibility, compactness, and randomness) The above Δ is invertible. In particular, the above Δ^{-1} is the inverse function. It is compact when ℓ_{str} is set to $\ell_{\mathsf{str}} \le BitLen(Dom(A))$. The output distribution of Δ is uniform over $\{0,1\}^{\ell_{\mathsf{str}} + \ell_{\mathrm{rec}}}$.

Lemma 4. (Collision resistance) If ℓ_{str} and ℓ_{rec} are polynomial in λ, the above Δ is collision resistant against polynomial-time adversaries that make a polynomial number of queries to $\mathcal{SE}$ and $\mathcal{SD}$. In particular, for any adversary asking q_d and q_e queries to $\mathcal{SE}$ and $\mathcal{SD}$, respectively, $\epsilon_{\Delta}^{\mathrm{coll}} \le \frac{q_d}{2^{\ell_{\mathsf{str}}}} + \frac{{q_e}^2}{2^{\ell_{\mathsf{str}}+\ell_{\mathrm{rec}}}}$ holds.

Proof. Observe that a collision happens only if

– there exists at least one pair of the same value among the returned values from $\mathcal{SE}$ or

– there exists a message returned from $\mathcal{SD}$ whose leading part is str.

The probability of the former case is upper bounded by $\frac{{q_e}^2}{2^{\ell_{\mathsf{str}}+\ell_{\mathrm{rec}}}}$ as a result of the birthday paradox among at most q_e randomly selected return values from $\mathcal{SE}$. Since every value returned from $\mathcal{SD}$ is random, the probability that the second case occurs when at most q_d queries are asked is at most $\frac{q_d}{2^{\ell_{\mathsf{str}}}}$. Summing up these bounds gives the upper bound as stated.

From these lemmas, recommendable settings would be $\ell_{\mathsf{str}} = \ell_{\mathrm{rec}} = BitLen(Dom(A))/2$ to balance the efficiency of the message recovery property and unforgeability.

With this instantiation of the Δ function and the Schnorr identification scheme defined over a group over an elliptic-curve as the underlying sigma-protocol, the resulting message recovery signature scheme turns out to be the ECPV scheme presented in [12].

5. Conclusion and open problems

We presented a generic method that transforms a sigma-protocol into a message recovery signature scheme in the random oracle model with specific properties of the redundancy function. The framework allows one to build a new message recovery signature scheme in a modular fashion such that the sigma-protocol and the redundancy function are designed and analyzed in completely separate ways. Thus, one can now focus on designing the redundancy function with the shown properties and then automatically obtain a secure digital signature scheme with message recovery.

Two specific redundancy functions were shown and it was proved that one meets all the sufficient properties in the random oracle model while the other meets those in the ideal cipher model. Combined with a sigma-protocol in our framework, the first redundancy function yields a refined version of the ECAO message recovery signature scheme with a more understandable and convincing modular security proof. This can be regarded as an example that shows that our new framework can improve existing schemes by increasing their security. The second redundancy function yields an already known scheme, ECPV with another security proof. Thus, our framework gives another insightful explanation into an existing scheme and validates its design. These examples show the usefulness of our framework.

One of the remaining challenges is to construct the redundancy function without using idealized assumptions. Another challenge is to find another framework

that yields shorter signed messages. One essential question is whether the redundancy is unavoidable or not for any message recovery signature schemes. Another direction of research includes relaxing the requirements for the redundancy function or even finding necessary and sufficient conditions.

References

[1] M. Bellare and P. Rogaway, "The exact security of digital signatures—how to sign with RSA and Rabin," in U. Maurer, editor, Advances in Cryptology, EUROCRYPT '96, Vol. 1070 of Lecture Notes in Computer Science, pp. 399–416, Springer-Verlag, 1996.

[2] S. Goldwasser, S. Micali, and R. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," SIAM Journal on Computing, Vol. 17, No. 2, pp. 281–308, Apr. 1988.

[3] M. Bellare and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols," First ACM Conference on Computer and Communication Security, pp. 62–73, Association for Computing Machinery, 1993.

[4] K. Nyberg and R. A. Rueppel, "A new signature scheme based on the DSA giving message recovery," in Proceedings of the First ACM Conference on Computer and Communications Security, 1993.

[5] K. Nyberg and R. A. Rueppel, "Message recovery for signature schemes based on the discrete logarithm problem," in Alfredo De Santis, editor, Advances in Cryptology, EUROCRYPT '94, Vol. 950 of Lecture Notes in Computer Science, pp. 182–193, Springer-Verlag, 1995.

[6] K. Nyberg and R. A. Rueppel, "Message recovery for signature schemes based on the discrete logarithm problem," Designs, Codes and Cryptography, Vol. 7, No. 1–2, pp. 61–81, 1996.

[7] A. Miyaji, "A message recovery signature scheme equivalent to DSA over elliptic curves," in Advances in Cryptology, Asiacrypt '96, Vol. 1163 of Lecture Notes in Computer Science, pp. 1–14, Springer-Verlag, 1996.

[8] M. Abe and T. Okamoto, "A signature scheme with message recovery as secure as discrete logarithm," IEICE Transaction of Fundamentals of Electronic Communications and Computer Science, E84-A(1), Jan. 2001 (presented at ASIACRYPT '99).

[9] C. P. Schnorr, "Efficient signature generation for smart cards," Journal of Cryptology, Vol. 4, No. 3, pp. 239–252, 1991.

[10] D. Naccache and J. Stern, "Signing on a postcard," Financial Cryptography 2000, Vol. 1962 of Lecture Notes in Computer Science, pp. 121–135, Springer-Verlag, 2001.

[11] U.S. Department of Commerce, National Institute of Standards and Technology, "Digital signature standard," FIPS PUB 186-2, 2000.

[12] L. Pintsov and S. Vanstone, "Postal revenue collection in the digital age," in Financial Cryptography 2000, Vol. 1962 of Lecture Notes in Computer Science, pp. 105–120, Springer-Verlag, 2000.

[13] D. Brown and D. Johnson, "Formal security proofs for a signature scheme with partial message recovery," in CT-RSA 2001, Vol. 2020 of Lecture Notes in Computer Science, pp. 126–142, Springer-Verlag, 2001.

[14] ISO/IEC 9796-3: Information technology—Security techniques—Digital signature schemes giving message recovery—Part 3: Discrete logarithm based mechanisms, 2nd Edition, JTC 1/SC 27, 2006.

[15] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in A. M. Odlyzko, editor, Advances in Cryptology, CRYPTO '86, Vol. 263 of Lecture Notes in Computer Science, pp. 186–199, Springer-Verlag, 1987.

[16] R. Cramer, "Modular Design of Secure yet Practical Cryptographic Protocols," Ph.D. thesis, Aula der Universiteit, 1996.

[17] J. Garay, P. MacKenzie, and K. Yang, "Strengthening zero-knowledge protocols using signatures," in E. Biham, editor, Advances in Cryptology, Eurocrypt '03, Vol. 2656 of Lecture Notes in Computer Science, pp. 177–194, Springer-Verlag, 2003.
Full version available from IACR e-print archive 2003/037.

[18] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," Journal of Cryptology, Vol. 13, No. 3, pp. 339–360, 2000.

Masayuki Abe
Distinguished Research Scientist, Okamoto Research Laboratory, NTT Information Sharing Platform Laboratories.
He received the B.E. and M.E. degrees in electrical engineering from the Science University of Tokyo, Tokyo, in 1990 and 1992, respectively, and Ph.D. degree from the University of Tokyo, Tokyo, in 2002. He joined NTT Network Information Systems Laboratories in 1992 and engaged in the development of fast algorithms for cryptographic functions and their software/hardware implementation and the development of a software cryptographic library. He is a member of the Institute of Electronics, Information and Communication Engineers (IEICE) of Japan and the Information Processing Society of Japan.

Tatsuaki Okamoto
Research Fellow, Okamoto Research Laboratory, NTT Information Sharing Platform Laboratories.
He received the B.E., M.E., and Dr.Eng. degrees from the University of Tokyo, Tokyo, in 1976, 1978, and 1988, respectively. He is a member of IEICE of Japan and the International Association for Cryptologic Research.

Koutarou Suzuki
Research Scientist, Information Security Project, NTT Information Sharing Platform Laboratories.
He received the B.S., M.S., and Ph.D. degrees from the University of Tokyo, Tokyo, in 1994, 1996, and 1999, respectively. He joined NTT Information Sharing Platform Laboratories in 1999. He has been engaged in research on public key cryptography, especially on cryptographic protocols and digital signatures. He is a member of IEICE and the Information Processing Society of Japan. He received the SCIS Paper Award from IEICE in 2002.