DE GRUYTER OPEN

TATRA MOUNTAINS
Mathematical Publications

DOI: 10.1515/tmmp-2017-0022
Tatra Mt. Math. Publ. **70** (2017), 119–138

GENERALIZED KEY SUBSTITUTION ATTACKS ON MESSAGE RECOVERY SIGNATURES

Atsushi Fujioka

ABSTRACT. The generalized key substitution attacks were proposed as a generalization of the key substitution attacks to examine the security of the signature schemes adopted in ISO/IEC (1st CD) 14888–3, which standardizes appendix-type signature schemes based on the discrete logarithm problem.

This paper examines the message recovery signature schemes based on the discrete logarithm problem, adopted in ISO/IEC 9796–3:2006, and shows that all but one scheme are vulnerable to the generalized key substitution attacks.

1. Introduction

Digital signatures are powerful tools to realize authenticity and integrity in the cyber world, and are applied to various protocols as primitives. A large variety of security notions are required because wide areas are covered by digital signatures. Discussing their security is one of the important topics in cryptography, to say nothing of network security.

In a digital signature scheme, an entity, called *signer*, has a private-key and can generate a signature for each message. Another entity, called *verifier*, has a public-key related to the private-key and can verify the validity of the signature of the message.

The security notions of digital signature schemes have been proposed and come to fruition as *existential unforgeability against chosen message attacks* (the EUF-CMA security). The EUF-CMA security is defined as a game between a challenger and an adversary who outputs a new pair of a message and its signature after getting many signatures on messages chosen by the adversary. The adversary wins the game when the outputted one is a valid, i.e., forged, pair containing a message which the adversary has never chosen. In this game,

2010 Mathematics Subject Classification: 94A60.
Keywords: message recovery signature, discrete logarithm, generalized key substitution attacks.

the adversary is allowed to determine the forgery message after seeing the public key. When the probability that the adversary wins the game is very low, the scheme is called *EUF-CMA secure*.

In a conventional scenario, a signature is handled with its message, and the validity is checked with the whole message and the signature. This type of signatures is called *digital signatures with message appendix* or *appendix-type signatures*. The appendix-type signatures are standardized in ISO/IEC 14888. ISO/IEC 14888–2 [6] and ISO/IEC 14888–3 [8] specify mechanisms based on integer factorization and on discrete logarithm, respectively. The *integer factorization problem* and the *discrete logarithm problem* are famous computational complexity problems in the number theory. Roughly speaking, the former is dividing a given composite number into prime factors, and the latter is computing a logarithm, x, from given y and g, where $y = g^x$.

1.1. Substitution Attacks

It is well known that *key exchange*, like the Diffie-Hellman protocol, provides secure key sharing but is vulnerable to the man-in-the-middle attack, that is, key exchange does not have key authentication. Digital signatures play an important role in *authenticated key agreement*. An authenticated key agreement protocol assures that host parties can share the key and adversaries can not guess it.

From such point of view, another security framework was proposed by Blake-Wilson and Menezes [3]. They considered a situation that a challenger provides pairs of messages and their signatures to an adversary and the adversary outputs new public-key (and its private key), where the message-signature pair is valid even under the new public-key. Blake-Wilson and Menezes noticed this situation when they analyzed the security of an authenticated key agreement protocol. When a digital signature scheme has such property, the security of the resultant authenticated key agreement protocol is harmed by the attack.

This scenario was called *duplicate-signature key selection attacks* in [3], and later, was named *key substitution attacks* (KS attacks) in [12]. Menezes and Smart formulate two security models:

- *strong* key substitution attacks (SKS attacks) and
- *weak* key substitution attacks (WKS attacks) [12].

In the SKS attacks, the adversary tries to output a substituted public-key under which a existing pair of a message and its signature is valid, and in the WKS attacks, it is necessary for the adversary to output a substituted pair of a public-key and the corresponding private-key. It is clear that the security against the SKS attacks implies the security against the WKS attacks, and we will state this formally in this paper.

Bohli, Röhrich, and Steinwandt defined a stronger security model on the KS attacks [4]. In their definition, the adversary tries to output two public-keys as they consider the security against malicious signers. Thus, we call their attacks *malicious signer key substitution attacks* (malicious signer KS attacks).

Blake-Wilson and Menezes showed that the RSA (-FDH, -PSS), Rabin, ElGamal, DSA, and ECDSA signatures are vulnerable to the KS attacks [3]. Menezes and Smart pointed out that the GHR-FDH signature is also vulnerable to the KS attacks however the Schnorr, DSA, ECDSA, and RW-FDH signatures have provable security under some assumptions and in some security models [3, 4, 12]. Bohli, Röhrich, and Steinwandt showed that the BB and NTRUSign signatures are vulnerable to malicious signer KS attacks [4]. See [3,4,12] in details.

It is worth to note here that another direction of the substitution attacks exists. Vaudenay considered the *domain parameter shifting attacks* [16,17], and discussed importance of the security against these attacks. In a signature scheme, all entities may agree with a parameter which is commonly used. The parameter is called *domain parameter.* In the domain parameter shifting attacks, later called the *domain parameter substitution attacks* in [4], the goal of the adversary is to output a new valid domain parameter under which the message and signature pair is valid.

Recently, Zhang, Yang, Zhang, and Chen extended the KS attacks and named them *generalized key substitution attacks* (GKS attacks) [19]. Roughly speaking, the adversary in the generalized *strong* key substitution attacks (GSKS attacks) tries to output a substituted public-key where the public-key has a form, (g, PK), g is a generator and PK is the other part of public-key, and the adversary in the generalized *weak* key substitution attacks (GWKS attacks) needs to output a substituted public-key, (g, PK), and the corresponding private-key. In other words, the generator is included in the public-key rather than the domain parameter.

Zhang et al. showed that all the signature schemes described in ISO/IEC 1st CD 14888–3 [7] are vulnerable to the G(W)KS attacks in a situation, where an adversary can manipulate both g and PK [19]. ISO/IEC 1st CD 14888–3 is a draft of the international standard specifying appendix-type signature schemes based on the discrete logarithm problem. The standard (draft) states that g (and other domain parameter) is selected by a user, and thus there is a possibility that both g and PK are changed. Note that the KS attacks are applicable to any signature schemes while the GKS attacks are effective only to signature schemes based the discrete logarithm problem. Actually, signature schemes based on the integer factorization problem, such as the RSA-FDH, RSA-PSS and GHR--FDH signatures, are vulnerable to the KS attacks [3, 12].

1.2. Message Recovery Signature

As introduced above, only the validity is checked in the appendix-type signature scenario when the signature is verified. In other words, the signature and the whole message are necessary to check the validity.

However, there exists another scenario. In the scenario, (a part of) a message is recovered from its signature when the signature is valid. This type of signatures is called *digital signatures with message recovery* or *message recovery signatures* (MRSs). In a MRS scheme, a message is split into two parts: *recovery* part and *clear* part. A signature is generated with the only recovery part of the message or with both the recovery and clear parts. The signature is handled with the clear part of the message, and the recovery part is computed from the signature when the signature is valid (with the clear part if necessary). MRSs have an advantage regarding the data size rather than appendix-type signatures, because we need not store the recover parts of messages. It is enough to store only signatures when the clear parts of the messages do not exist.

The MRSs are standardized in ISO/IEC 9796. ISO/IEC 9796–2 [9] and ISO/IEC 9796–3 [10] specify mechanisms based on integer factorization and on discrete logarithm, respectively.

1.3. Contributions

In this paper, we propose definitions of the KS attacks and the GKS attacks on MRSs. We formally state relations among them including their strong and weak variants, and explain what the relations show.

Next, we examine vulnerability of the MRS schemes adopted in ISO/IEC 9796--3:2006 [10] against the GWKS attacks. ISO/IEC 9796–3:2006 describes six MRS schemes:

- Nyberg-Rueppel MRS (NR),
- elliptic curve Nyberg-Rueppel MRS (ECNR),
- elliptic curve Miyaji MRS (ECMR),
- elliptic curve Abe-Okamoto MRS (ECAO),
- elliptic curve Pintsov-Vanstone MRS (ECPV), and
- elliptic curve KCDSA/Nyberg-Rueppel MRS (ECKNR).

We show that NR, ECNR, ECMR, ECAO, and ECPV are vulnerable to the GWKS attack, and discuss the security of ECKNR against the GWKS attacks. Note that this standard also states that g (and other domain parameter) is selected by a user, and thus there is a possibility that both g and PK are changed.

Organization

Section 2 provides definitions of MRS schemes and KS attacks on MRS schemes. We introduce the MRS schemes adopted in ISO/IEC 9796–3:2006 in Section 3 and discuss the security of them against the GWKS attacks in Section 4. Section 5 concludes the paper, and relations among the KS attacks are shown in Appendix.

2. Definitions

In this section, we provide definitions of MRS schemes and the KS attacks on MRS schemes.

Let $\mathbb{G}$ be a cyclic group of a prime order, q. Note that any element Y in $\mathbb{G}$ can be expressed with generator G as $Y = G^x$.

Throughout this paper, $\mathbb{Z}_n$ is the ring of integers modulo n, and a message, m, given as an octet string, is divided into m_{rec} and m_{clr}, i.e., $m = m_{\text{rec}}||m_{\text{clr}}$, where $||$ is a concatenation of octet strings.

Definition 1 (Discrete Logarithm Problem)**.** For a randomly chosen element, $Y \in \mathbb{G}$, and a random generator, $G \in \mathbb{G}$, the *discrete logarithm problem* is, given $(G, Y) \in \mathbb{G}^2$, to compute element $x \in \mathbb{Z}_q$, where $Y = G^x$.

This definition is expressed using a multiplicative group. However, it can be expressed in an additive group as well.

Next, we define MRSs. In a MRS scheme, message m consists of recovery part m_{rec} and clear part m_{clr}, i.e., $m = m_{\text{rec}}||m_{\text{clr}}$. The clear part is handled with a signature to retrieve the recovery part. Note that it is allowed that the clear part does not exist, i.e., m_{clr} is the null string.

Definition 2 (Message Recovery Signature Scheme)**.** A *message recovery signature* scheme is a tuple of probabilistic polynomial-time algorithms ($\mathcal{D}$, $\mathcal{D}_V$, $\mathcal{G}$, $\mathcal{G}_V$, $\mathcal{S}$, $\mathcal{S}_V$) as follows:

- $\mathcal{D}(1^\kappa)$*: The algorithm takes* 1^κ *as the input, where* κ *is a security parameter. It outputs domain parameter params.*
- $\mathcal{D}_V(params)$*: The algorithm takes params as the input. It outputs* `accept` *if params is valid. Otherwise, it outputs* `reject`.
- $\mathcal{G}(params)$*: The algorithm takes domain parameter params as the input. It outputs public-key* PK *and private-key* SK.
- $\mathcal{G}_V(params, PK)$*: The algorithm takes params and public-key* PK *as inputs. It outputs* `accept` *if* PK *is valid under domain parameter params. Otherwise, it outputs* `reject`.

- $\mathcal{S}(params, SK, m)$: *The algorithm takes domain parameter params, private-key SK, and message m* $(= m_{\mathrm{rec}}||m_{\mathrm{clr}})$ *as inputs. It outputs signature* σ *on* m.
- $\mathcal{S}_V(params, PK, m_{\mathrm{clr}}, \sigma)$: *The algorithm takes params, public-key PK, the clear part of the message,* m_{clr}, *and signature* σ *as inputs. It outputs* m $(= m_{\mathrm{rec}}||m_{\mathrm{clr}})$ *if* σ *is a signature on* m. *Otherwise, it outputs* `reject`.

These algorithms are assumed to satisfy

$$\mathcal{S}_V\big(params, PK, m_{\mathrm{clr}}, \mathcal{S}(params, SK, m)\big) = m$$

for any m $(= m_{\mathrm{rec}}||m_{\mathrm{clr}})$, for any *params* generated by the $\mathcal{D}$ algorithm, and for any PK, SK generated by the $\mathcal{G}$ algorithm with *params*, where $\mathcal{D}_V(params) =$ `accept` and $\mathcal{G}_V(params, PK) =$ `accept` hold.

When we define the weak attacks, we need to change $\mathcal{G}_V$ to a zero-knowledge interactive algorithm, $\mathcal{G}_V((params, PK, SK), (params, PK))$, where a *prover* has PK, SK and *params*, and a *verifier* has *params* and PK. $\mathcal{G}_V$ allows the prover to demonstrate to the verifier that PK is a valid public-key corresponding to the private-key SK in the zero-knowledge manner [11]. The interactive algorithm outputs `accept` if the prover can demonstrate it. Otherwise, it outputs `reject`.

Note that the standard [10] does not specify $\mathcal{D}_V(params)$ and $\mathcal{G}_V(params, PK)$ explicitly but requires such algorithms as "the signer may also require assurance that the domain parameters and public verification key are valid, otherwise an adversary may be able to generate signatures that verify."

The KS attacks and its generalization have two types of attack scenarios: strong and weak. Thus, we have four types of the KS attacks: the strong KS attacks, the WKS attacks, the GSKS attacks, and the GWKS attacks.

Next, we define the security against the KS attacks.

The security of $\mathcal{MRS}$ is defined on the following experiment, $\mathbf{Exp}^{\mathsf{atk}}_{\mathcal{MRS},\mathcal{A}}(\kappa)$, between a challenger and an adversary, $\mathcal{A}$, where atk denotes a type of attacks such that $\mathsf{atk} \in \{\mathsf{sksa}, \mathsf{wksa}\}$.[1]

Experiment $\mathbf{Exp}^{\mathsf{atk}}_{\mathcal{MRS},\mathcal{A}}(\kappa)$**:**

Setup Phase: The challenger obtains $params \leftarrow \mathcal{D}(1^\kappa)$, then generates $(PK, SK) \leftarrow \mathcal{G}(params)$, and initializes $MS \leftarrow \emptyset$, where MS denotes the set of message and signature pairs. The adversary, $\mathcal{A}$, is given domain parameter *params* and public-key PK.

Learning Phase: $\mathcal{A}$ can query the Sign oracle.

- The oracle, Sign, receives input m $(= m_{\mathrm{rec}}||m_{\mathrm{clr}})$. It computes σ, a signature on m, as $\sigma, \leftarrow \mathcal{S}(params, SK, m)$ and returns σ. The oracle adds $(m_{\mathrm{clr}}, \sigma)$ to MS.

[1]Although the security definition in [12] combines the EUF-CMA security and the security against KS attacks, we define the security against KS attacks only like in [4, 19].

Challenge Phase: $\mathcal{A}$ sends a target public-key, $\widetilde{PK}$, a clear part of a message, $\tilde{m}_{\rm clr}$ and a signature, $\tilde{\sigma}$, to the challenger. $\mathcal{A}$ sends a target private-key, $\widetilde{SK}$, also if $\mathsf{atk} = \mathsf{wksa}$.
The challenger in the case of $\mathsf{atk} = \mathsf{sksa}$ outputs 1 as the result of this experiment if $PK \neq \widetilde{PK} \wedge (\tilde{m}_{\rm clr}, \tilde{\sigma}) \in MS \wedge \mathcal{G}_V(params, \widetilde{PK}) = \mathtt{accept} \wedge \mathcal{S}_V(params, \widetilde{PK}, \tilde{m}_{\rm clr}, \tilde{\sigma}) = \tilde{m}$, where $\tilde{m} = \tilde{m}_{\rm rec}||\tilde{m}_{\rm clr}$. Otherwise, 0. In the case of $\mathsf{atk} = \mathsf{wksa}$, the key validation condition, $\mathcal{G}_V(params, \widetilde{PK}) = \mathtt{accept}$, is changed to a $\mathcal{G}_V((params, \widetilde{PK}, \widetilde{SK}), (params, \widetilde{PK})) = \mathtt{accept}$.

Definition 3 (Security against Key Substitution Attacks)**.** Let $\mathcal{MRS} = (\mathcal{D}, \mathcal{D}_V, \mathcal{G}, \mathcal{G}_V, \mathcal{S}, \mathcal{S}_V)$ be a MRS scheme and $\mathcal{A}$ an adversary. Let κ be a security parameter. The advantage of $\mathcal{A}$ in attacking $\mathcal{MRS}$ is defined by

$$\mathbf{Adv}^{\mathsf{atk}}_{\mathcal{MRS},\mathcal{A}}(\kappa) := \Pr\left[\ \mathbf{Exp}^{\mathsf{atk}}_{\mathcal{MRS},\mathcal{A}}(\kappa) = 1\ \right],$$

where $\mathsf{atk} \in \{\mathsf{sksa}, \mathsf{wksa}\}$. We say that $\mathcal{MRS}$ is *secure against strong key substitution attacks (*SKSA *secure)* if $\mathbf{Adv}^{\mathsf{sksa}}_{\mathcal{MRS},\mathcal{A}}(\kappa)$ is negligible for every polynomial-time $\mathcal{A}$ and is *secure against weak key substitution attacks (*WKSA *secure)* if $\mathbf{Adv}^{\mathsf{wksa}}_{\mathcal{MRS},\mathcal{A}}(\kappa)$ is negligible for every polynomial-time $\mathcal{A}$.

We formulate the following experiment, $\mathbf{Exp}^{\mathsf{atk}}_{\mathcal{MRS},\mathcal{A}}(\kappa)$, between a challenger and an adversary, $\mathcal{A}$, where atk denotes a type of attacks such that $\mathsf{atk} \in \{\mathsf{gsksa}, \mathsf{gwksa}\}$ in the similar way. In the experiment, we consider a MRS scheme based on the discrete logarithm problem, and restrict the scheme, where public-key PK has a form, $PK = (g, \textsc{pk})$, where g is a generator, $\textsc{pk} = g^x$, and x is the private-key, SK.

We omit **Setup Phase** and **Learning Phase** as they are identical with the phases in the above experiment.

Experiment $\mathbf{Exp}^{\mathsf{atk}}_{\mathcal{MRS},\mathcal{A}}(\kappa)$:

Challenge Phase: $\mathcal{A}$ sends a target generator and a public key, $(\tilde{g}, \widetilde{\textsc{pk}})$, a message, $\tilde{m}$ and a forged signature, $\tilde{\sigma}$, to the challenger. $\mathcal{A}$ sends a target private-key, $\widetilde{SK}$, also if $\mathsf{atk} = \mathsf{gwksa}$.
The challenger in the case of $\mathsf{atk} = \mathsf{gsksa}$ outputs 1 as the result of this experiment if $(g, \textsc{pk}) \neq (\tilde{g}, \widetilde{\textsc{pk}}) \wedge (\tilde{m}_{\rm clr}, \tilde{\sigma}) \in MS \wedge \mathcal{G}_V(params, (\tilde{g}, \widetilde{\textsc{pk}})) = \mathtt{accept} \wedge \mathcal{S}_V(params, (\tilde{g}, \widetilde{\textsc{pk}}), \tilde{m}_{\rm clr}, \tilde{\sigma}) = \tilde{m}$, where $\tilde{m} = \tilde{m}_{\rm rec}||\tilde{m}_{\rm clr}$. Otherwise, 0. In the case of $\mathsf{atk} = \mathsf{gwksa}$, the key validation condition, $\mathcal{G}_V(params, (\tilde{g}, \widetilde{\textsc{pk}}))=\mathtt{accept}$, is changed to $\mathcal{G}_V((params, (\tilde{g}, \widetilde{\textsc{pk}}), \widetilde{SK}), (params, (\tilde{g}, \widetilde{\textsc{pk}})))=\mathtt{accept}$.

Definition 4 (Security against Generalized Key Subsitution Attacks)**.** Let $\mathcal{MRS} = (\mathcal{D}, \mathcal{D}_V, \mathcal{G}, \mathcal{G}_V, \mathcal{S}, \mathcal{S}_V)$ be a MRS scheme based on a discrete logarithm problem and $\mathcal{A}$ an adversary. Let κ be a security parameter. The advantage of $\mathcal{A}$ in attacking $\mathcal{MRS}$ is defined by

$$\mathbf{Adv}^{\mathsf{atk}}_{\mathcal{MRS},\mathcal{A}}(\kappa) := \Pr\left[\ \mathbf{Exp}^{\mathsf{atk}}_{\mathcal{MRS},\mathcal{A}}(\kappa) = 1\ \right],$$

where $\mathsf{atk} \in \{\mathsf{gsksa}, \mathsf{gwksa}\}$. We say that $\mathcal{MRS}$ is *secure against generalized strong key substitution attacks (*GSKSA *secure)* if $\mathbf{Adv}^{\mathsf{gsksa}}_{\mathcal{MRS},\mathcal{A}}(\kappa)$ is negligible for every polynomial-time $\mathcal{A}$ and is *secure against generalized weak key substitution attacks (*GWKSA *secure)* if $\mathbf{Adv}^{\mathsf{gwksa}}_{\mathcal{MRS},\mathcal{A}}(\kappa)$ is negligible for every polynomial--time $\mathcal{A}$.

We show the relations among the security notions in Appendix.

3. MRS Schemes in ISO/IEC 9796–3:2006

In this section, we introduce the MRS schemes adopted in ISO/IEC 9796--3:2006 [10].

3.1. NR

In this subsection, we describe the Nyberg-Rueppel MRS scheme (NR). NR appeared in [14].

$\mathcal{D}(1^\kappa)$**:** It generates a multiplicative group, $\mathbb{G}$, of a finite field, $\mathbb{F}$, where $\mathbb{G}$ is generated by a generator, G, whose order is a prime, n, depending on security parameter κ, and outputs $params = (G, n, \mathbb{G}, \mathbb{F})$.

$\mathcal{D}_V(params)$**:** It checks that $\mathbb{F}$ is a finite field depending on security parameter κ, $\mathbb{G}$ is a multiplicative group of $\mathbb{F}$, and $\mathbb{G}$ is generated by a generator, G, whose order is a prime, n. It outputs `accept` if the checks were passed. Otherwise, it outputs `reject`.

$\mathcal{G}(params)$**:** It randomly generates private-key $x \in \mathbb{Z}_n^*$, computes public-key $Y = G^x$, and then, outputs (G, Y) as PK and x as SK, respectively.

$\mathcal{G}_V((params, PK, SK), (params, PK))$**:** It outputs `accept` if the prover can show possession of x (on Y) to the verifier. Otherwise, it outputs `reject`.

$\mathcal{S}(params, SK, m)$**:** It generates a random number, $k \in \mathbb{Z}_n$, computes $R = G^k$, and converts the field element, R, to pre-signature Π which is an octet string. It converts the octet string, Π, to an integer, π, computes $\rho = (\delta + \pi) \bmod n$ and $s = (k - x\rho) \bmod n$, where δ is a integer converted from m_{rec}, converts the integer, ρ, to an octet string, r, and then outputs (r, s) as a signature, σ $(= (r, s))$. If $\rho = 0$ or $s = 0$ happen, it recomputes a signature with newly generated k.

$\mathcal{S}_V(params, PK, m_{\rm clr}, \sigma)$**:** It checks that r' is an octet string with the proper length and that $s' \in \mathbb{Z}_n^*$ hold, where $\sigma = (r', s')$. If not, it outputs `reject`. It converts the octet string, r', to an integer, ρ', computes $R' = G^{s'}Y^{\rho'}$, and converts the field element, R', to an integer, π' (via recovered pre-signature Π' which is an octet string). Then, it computes $\delta' = (\rho' - \pi') \bmod n$, converts the integer, δ', to an octet string, d', and regards d' as $m'_{\rm rec}$.

After the redundancy check of $m'_{\rm rec}$, it outputs the recovered message, m' $(= m'_{\rm rec}||m_{\rm clr})$, if it passed the check. Otherwise, it outputs `reject`.

Note that the clear part of the message, $m_{\rm clr}$, is not used in $\mathcal{S}$ and $\mathcal{S}_V$. So we may omit $m_{\rm clr}$ from the inputs.

$\mathcal{G}_V$ is not specified in the standard. However, a zero-knowledge proof for possession of the discrete logarithm such as in [5] can be used. For the other schemes, the same discussion is applied.

3.2. ECNR

In this subsection, we describe a MRS scheme constructed on an elliptic curve. The scheme is based on NR, and is called the elliptic curve Nyberg-Rueppel MRS scheme (ECNR).

$\mathcal{D}(1^\kappa)$**:** It generates an additive group, $\mathbb{G}$, of a finite field, $\mathbb{F}$, where $\mathbb{G}$ is generated by a generator, G, whose order is a prime, n, depending on security parameter κ, and outputs $params = (G, n, \mathbb{G}, \mathbb{F})$.

$\mathcal{D}_V(params)$**:** It checks that $\mathbb{F}$ is a finite field depending on security parameter κ, $\mathbb{G}$ is an additive group of $\mathbb{F}$, and $\mathbb{G}$ is generated by a generator, G, whose order is a prime, n. It outputs `accept` if the checks were passed. Otherwise, it outputs `reject`.

$\mathcal{G}(params)$**:** It randomly generates private-key $x \in \mathbb{Z}_n^*$, computes public-key $Y = xG$, and then, outputs (G, Y) as PK and x as SK, respectively.

$\mathcal{G}_V((params, PK, SK), (params, PK))$**:** It outputs `accept` if the prover can show possession of x (on Y) to the verifier. Otherwise, it outputs `reject`.

$\mathcal{S}(params, SK, m)$**:** It generates a random number, $k \in \mathbb{Z}_n$, computes $R = kG$, and converts the field element, R, to an integer, π (via recovered pre-signature Π' which is an octet string). It computes $\rho = (\delta + \pi) \bmod n$ and $s = (k - x\rho) \bmod n$, converts the integer, ρ, to an octet string, r, and then outputs (r, s) as a signature, σ $(= (r, s))$. If $\rho = 0$ or $s = 0$ happen, it recomputes a signature with newly generated k.

$\mathcal{S}_V(params, PK, m_{\rm clr}, \sigma)$**:** It checks that r' is an octet string with the proper length and that $s' \in \mathbb{Z}_n^*$ hold, where $\sigma = (r', s')$. If not, it outputs `reject`. It converts the octet string, r' to an integer, ρ', computes R' as $R' = s'G + \rho'Y$, and converts the field element, R', to an integer, π' (via recovered pre-signature Π' which is an octet string). It computes $\delta = (\rho' - \pi') \bmod n$, converts the integer, δ, to an octet string, d, and then, regards d' as m'_{rec}.

After the redundancy check of m'_{rec}, it outputs the recovered message, m' $(= m'_{\text{rec}} || m_{\text{clr}})$, if it passed the check. Otherwise, it outputs `reject`.

Note that the clear part of the message, m_{clr}, is not used in $\mathcal{S}$ and $\mathcal{S}_V$.

3.3. ECMR

In this subsection, we describe the elliptic curve Miyaji MRS scheme (ECMR). ECMR appeared in [13].

$\mathcal{D}(1^\kappa)$**:** It generates an additive group, $\mathbb{G}$, of a finite field, $\mathbb{F}$, where $\mathbb{G}$ is generated by a generator, G, whose order is a prime, n, depending on security parameter κ, and outputs $\mathit{params} = (G, n, \mathbb{G}, \mathbb{F})$.

$\mathcal{D}_V(\mathit{params})$**:** It checks that $\mathbb{F}$ is a finite field depending on security parameter κ, $\mathbb{G}$ is an additive group of $\mathbb{F}$, and $\mathbb{G}$ is generated by a generator, G, whose order is a prime, n. It outputs `accept` if the checks were passed. Otherwise, it outputs `reject`.

$\mathcal{G}(\mathit{params})$**:** It randomly generates private-key $x \in \mathbb{Z}_n^*$, computes public-key $Y = xG$, and then, outputs (G, Y) as PK and x as SK, respectively.

$\mathcal{G}_V((\mathit{params}, PK, SK), (\mathit{params}, PK))$**:** It outputs `accept` if the prover can show possession of x (on Y) to the verifier. Otherwise, it outputs `reject`.

$\mathcal{S}(\mathit{params}, SK, m)$**:** It generates a random number, $k \in \mathbb{Z}_n$, computes $R = kG$, and computes Π as $\Pi = \text{Mask}(R)$, where Mask is a masking function. It computes $r = d \oplus \Pi$, where $d = m_{\text{rec}}$, converts the octet string, r, to an integer, ρ, computes $s = \frac{\rho k - \rho - 1}{x + 1} \bmod n$, and then outputs (r, s) as a signature, σ $(= (r, s))$. If $\rho = 0$ or $s = 0$ happen, it recomputes a signature with newly generated k.

$\mathcal{S}_V(\mathit{params}, PK, m_{\text{clr}}, \sigma)$**:** It checks that r' is an octet string with the proper length and that $s' \in \mathbb{Z}_n^*$ hold, where $\sigma = (r', s')$. If not, it outputs `reject`. It converts the octet string, r', to an integer, ρ', computes $R' = \frac{1+\rho'+s'}{\rho'} G + \frac{s'}{\rho'} Y$, $\Pi' = \text{Mask}(R')$, and $d' = r' \oplus \Pi'$, and then, regards d' as m'_{rec}.

After the redundancy check of m'_{rec}, it outputs the recovered message, m' $(= m'_{\text{rec}} || m_{\text{clr}})$, if passed the check. Otherwise, it outputs `reject`.

Note that the clear part of the message, m_{clr}, is not used in $\mathcal{S}$ and $\mathcal{S}_V$.

3.4. ECAO

In this subsection, we describe the elliptic curve Abe-Okamoto MRS scheme (ECAO). ECAO appeared in [1].

$\mathcal{D}(1^\kappa)$**:** It generates an additive group, $\mathbb{G}$, of a finite field, $\mathbb{F}$ where $\mathbb{G}$ is generated by a generator, G, whose order is a prime, n, depending on security parameter κ, and outputs $params = (G, n, \mathbb{G}, \mathbb{F})$.

$\mathcal{D}_V(params)$**:** It checks that $\mathbb{F}$ is a finite field depending on security parameter κ, $\mathbb{G}$ is an additive group of $\mathbb{F}$, and $\mathbb{G}$ is generated by a generator, G, whose order is a prime, n. It outputs `accept` if the checks were passed. Otherwise, it outputs `reject`.

$\mathcal{G}(params)$**:** It randomly generates private-key $x \in \mathbb{Z}_n^*$, computes public-key $Y = xG$, and then, outputs (G, Y) as PK and x as SK, respectively.

$\mathcal{G}_V((params, \mathbf{PK}, SK), (params, PK))$**:** It outputs `accept` if the prover can show possession of x (on Y) to the verifier. Otherwise, it outputs `reject`.

$\mathcal{S}(params, SK, m)$**:** It generates a random number, $k \in \mathbb{Z}_n$, computes $R = kG$, and converts the field element, R, to an octet string, Π. It computes $r = d \oplus \Pi$, where d is an octet string generated from m with the OAEP-type conversion [2]. It computes $u = \mathrm{MGF}(r||m_{\mathrm{clr}})$, where MGF is a mask generation function, converts the octet string, u, to an integer, t, computes $s = (k - xt) \bmod n$, and then outputs (r, s) as a signature, σ $(= (r, s))$. If $s = 0$ or $t = 0$ happen, it recomputes a signature with newly generated k.

$\mathcal{S}_V(params, PK, m_{\mathrm{clr}}, \sigma)$**:** It checks that r' is an octet string with the proper length and that $s' \in \mathbb{Z}_n^*$ hold, where $\sigma = (r', s')$. If not, it outputs `reject`. It computes $u' = \mathrm{MGF}(r'||m'_{\mathrm{clr}})$, converts the octet string, u', to an integer, t', and computes $R' = s'P + t'Q$. If R' the zero point of the finite field, it outputs `reject`. It converts the field element, R', to an octet string, Π', and computes $d' = r' \oplus \Pi'$.

After the OAEP-type redundancy check of d' with m'_{clr}, it outputs the recovered message, m' $(= m'_{\mathrm{rec}}||m_{\mathrm{clr}})$, if it passed the check. Otherwise, it outputs `reject`.

Note that the clear part of the message, m_{clr}, is used in $\mathcal{S}$ and $\mathcal{S}_V$.

3.5. ECPV

In this subsection, we describe the elliptic curve Pintsov-Vanstone MRS scheme (ECPV). ECPV appeared in [15].

$\mathcal{D}(1^\kappa)$**:** It generates an additive group, $\mathbb{G}$, of a finite field, $\mathbb{F}$ where $\mathbb{G}$ is generated by a generator, G, whose order is a prime, n, depending on security parameter κ, and outputs $params = (G, n, \mathbb{G}, \mathbb{F})$.

$\mathcal{D}_V(params)$**:** It checks that $\mathbb{F}$ is a finite field depending on security parameter κ, $\mathbb{G}$ is an additive group of $\mathbb{F}$, and $\mathbb{G}$ is generated by a generator, G, whose order is a prime, n. It outputs `accept` if the checks were passed. Otherwise, it outputs `reject`.

$\mathcal{G}(params)$: It randomly generates private-key $x \in \mathbb{Z}_n^*$, computes public-key $Y = xG$, and then, outputs (G, Y) as PK and x as SK, respectively.

$\mathcal{G}_V((params, PK, SK), (params, PK))$: It outputs `accept` if the prover can show possession of x (on Y) to the verifier. Otherwise, it outputs `reject`.

$\mathcal{S}(params, SK, m)$: It generates a random number, $k \in \mathbb{Z}_n$, computes $R = kG$, and computes $\Pi = \mathrm{KDF}(x\text{-coordinate of } R)$, $r = \mathrm{Sym}(d, \Pi)$, $u = \mathrm{Hash}(r \| m_{\mathrm{clr}})$, where KDF is a key derivation function, Sym is the encryption algorithm of a symmetric cipher, Hash is an hash function, and $d = m_{\mathrm{rec}}$. It converts the octet string, u, to an integer, t, computes $s = (k - xt)$, and then outputs (r, s) as a signature, σ $(= (r, s))$. If $s = 0$ or $t = 0$ happen, it recomputes a signature with newly generated k.

$\mathcal{S}_V(params, PK, m_{\mathrm{clr}}, \sigma)$: It checks that r' is an octet string with the proper length and that $s' \in \mathbb{Z}_n^*$ hold, where $\sigma = (r', s')$. If not, it outputs `reject`. It computes $u' = \mathrm{Hash}(r' \| m'_{\mathrm{clr}})$, converts the octet string. u', to an integer, t', computes $R' = s'P + t'Q$, $\Pi' = \mathrm{KDF}(x\text{-coordinate of } R')$, $d' = \mathrm{Sym}^{-1}(r', \Pi')$ where Sym is the decryption algorithm of the symmetric cipher, and then, regards d' as m'_{rec}.

After the redundancy check of m'_{rec}, it outputs the recovered message, m' $(= m'_{\mathrm{rec}} \| m_{\mathrm{clr}})$, if it passed the check. Otherwise, it outputs `reject`.

Note that the clear part of the message, m_{clr}, is not used in $\mathcal{S}$ and $\mathcal{S}_V$.

3.6. ECKNR

In this subsection, we describe the elliptic curve KCDSA/Nyberg-Rueppel MRS scheme (ECKNR). ECKNR appeared in [18].

We assume that any field element, P $(\in \mathbb{G})$, is expressed with x-coordinate of P and y-coordinate of P.

$\mathcal{D}(1^\kappa)$: It generates an additive group, $\mathbb{G}$, of a finite field, $\mathbb{F}$ where $\mathbb{G}$ is generated by a generator, G, whose order is a prime, n, depending on security parameter κ, and outputs $params = (G, n, \mathbb{G}, \mathbb{F})$.

$\mathcal{D}_V(params)$: It checks that $\mathbb{F}$ is a finite field depending on security parameter κ, $\mathbb{G}$ is an additive group of $\mathbb{F}$, and $\mathbb{G}$ is generated by a generator, G, whose order is a prime, n. It outputs `accept` if the checks were passed. Otherwise, it outputs `reject`.

$\mathcal{G}(params)$: It randomly generates private-key $x \in \mathbb{Z}_n^*$, computes public-key $Y = xG$, and then, outputs (G, Y) as PK and x as SK, respectively.

$\mathcal{G}_V((params, PK, SK), (params, PK))$: It outputs `accept` if the prover can show possession of x (on Y) to the verifier. Otherwise, it outputs `reject`.

$\mathcal{S}(params, SK, m)$: It generates a random number, $k \in \mathbb{Z}_n$, computes $R = kG$, and converts the field element, R, to pre-signature Π as $\Pi = \mathrm{MGF}(R)$

where MGF is a mask generation function. Note that Π is an octet string. It computes r as $r = d \oplus \Pi \oplus \mathrm{MGF}(z||m_{\mathrm{clr}})$ where $d = m_{\mathrm{rec}}$ and z is the certificate data of the public-key, Y, such as $z = [x$-coordinate of $Y||y$-coordinate of $Y]$. It computes s as $s = (k - x\rho) \bmod n$ where ρ is an integer converted from the octet string, r. Then, it outputs (r, s) as a signature, σ $(= (r, s))$. If $\rho = 0$ or $s = 0$ happen, it recomputes a signature with newly generated k.

$\mathcal{S}_V(params, PK, m_{\mathrm{clr}}, \sigma)$: It checks that r' is an octet string with the proper length and that $s' \in \mathbb{Z}_n^*$ hold, where $\sigma = (r', s')$. If not, it outputs `reject`. It converts the octet string, r' to an integer, ρ', computes R' as $R' = s'G + \rho'Y$, and converts R' to an octet string, Π'. It computes $d = r' \oplus \Pi' \oplus \mathrm{MGF}(z||m'_{\mathrm{clr}})$, where $z = [x$-coordinate of $Y||y$-coordinate of $Y]$, and then, regards d' as m'_{rec}.

After the redundancy check of m'_{rec}, it outputs the recovered message, m' $(= m'_{\mathrm{rec}}||m_{\mathrm{clr}})$, if it passed the check. Otherwise, it outputs `reject`.

4. GWKSA Security of MRS in ISO/IEC 9796–3:2006

In this section, we indicate that NR is vulnerable to the GWKS attack. Next, we provide an unified representation of ECNR, ECMR, ECAO, and ECPV, and show the GWKS attack on the representation. Finally, we discuss the security of ECKNR against the GWKS attacks.

4.1. GWKS attack on NR

We describe an adversary, $\mathcal{A}$, who obtains only single pair, (m, σ), and performs the GWKS attack on NR, where $m = m'_{\mathrm{rec}}||m_{\mathrm{clr}}$ and $\sigma = (r, s)$:

(1) $\mathcal{A}$ computes $\tilde{R} = G^s Y^\rho$ from the valid signature, σ $(= (r, s))$, and ρ is the converted integer from the octet string, r.

(2) $\mathcal{A}$ generates a random number, $\tilde{x} \in \mathbb{Z}_n^*$.

(3) $\mathcal{A}$ computes $\tilde{G} = \tilde{R}^{\frac{1}{s+\tilde{x}\rho}}$.

(4) $\mathcal{A}$ computes $\tilde{Y} = \tilde{G}^{\tilde{x}}$.

(5) $\mathcal{A}$ outputs $\tilde{G}$, $\tilde{Y}$, $\tilde{x}$, $\tilde{m}_{\mathrm{clr}}$ $(= m_{\mathrm{clr}})$, and $\tilde{\sigma}$ $(= \sigma)$.

We show that $\mathcal{S}_V$ outputs m $(m = m_{\mathrm{rec}}||m_{\mathrm{clr}})$ when the original signature, σ $(= (r, s))$ and the clear part of the original message, m_{clr}, are given to $\mathcal{S}_V$ with the substituted keys, $(\tilde{G}, \tilde{Y})$, as follows: Note that $G \neq \tilde{G}$, $Y \neq \tilde{Y}$, $m_{\mathrm{clr}} = \tilde{m}_{\mathrm{clr}}$, and $\sigma = \tilde{\sigma}$. $\mathcal{S}_V$ converts the octet string, r, to an integer, ρ, computes $\tilde{R}' = \tilde{G}^s \tilde{Y}^\rho$, converts the field element, $\tilde{R}'$, to an integer, $\tilde{\pi}'$ (via recovered pre-signature $\tilde{\Pi}'$ which is an octet string). Then, $\mathcal{S}_V$ computes $\tilde{\delta}' = (\rho - \tilde{\pi}') \bmod n$, converts the integer, $\tilde{\delta}'$, to an octet string, $\tilde{d}'$, and regards $\tilde{d}'$ as $\tilde{m}'_{\text{rec}}$. We have $\tilde{R}' = \tilde{G}^s\tilde{Y}^\rho = (\tilde{R}^{\frac{1}{s+\tilde{x}\rho}})^s(\tilde{G}^{\tilde{x}})^\rho = (\tilde{R}^{\frac{1}{s+\tilde{x}\rho}})^s(\tilde{R}^{\frac{\tilde{x}}{s+\tilde{x}\rho}})^\rho = (\tilde{R}^{\frac{1}{s+\tilde{x}\rho}})^{s+\tilde{x}\rho} = \tilde{R} = G^sY^\rho$. Thus, $m'_{\text{rec}} = \tilde{m}_{\text{rec}}$ holds because $\tilde{R}'$ and $\tilde{R}$ produce the same pre-signature, $\tilde{\Pi}'$ $(= \Pi')$, and $\tilde{m}'_{\text{rec}}$ are recovered from Π' and ρ in the same way with m'_{rec}.

Therefore, $\mathcal{A}$ can succeed in the GWKS attack.

4.2. GWKS attack on ECNR, ECMR, ECAO, and ECPV

In this subsection, we provide an unified representation of MRSs adopted in ISO/IEC 9796–3:2006. This representation can handle all MRS schemes but the elliptic curve KCDSA/Nyberg-Rueppel MRS scheme (ECKNR). We show the GWKS attack on them.

4.2.1. General Framework

We provide an unified representation of ECNR, ECMR, ECAO, and ECPV.

$\mathcal{D}(1^\kappa)$: It generates an additive group, $\mathbb{G}$, of a finite field, $\mathbb{F}$ where $\mathbb{G}$ is generated by a generator, G, whose order is a prime, n, depending on security parameter κ, and outputs $\mathit{params} = (G, n, \mathbb{G}, \mathbb{F})$.

$\mathcal{D}_V(\mathit{params})$: It checks that $\mathbb{F}$ is a finite field depending on security parameter κ, $\mathbb{G}$ is an additive group of $\mathbb{F}$, and $\mathbb{G}$ is generated by a generator, G, whose order is a prime, n. It outputs `accept` if the checks were passed. Otherwise, it outputs `reject`.

$\mathcal{G}(\mathit{params})$: It randomly generates private-key $x \in \mathbb{Z}_n^*$, computes public-key $Y = xG$, and then, outputs (G, Y) as PK and x as SK, respectively.

$\mathcal{G}_V((\mathit{params}, PK, SK), (\mathit{params}, PK))$: It outputs `accept` if the prover can show possession of x (on Y) to the verifier. Otherwise, it outputs `reject`.

$\mathcal{S}(\mathit{params}, SK, m)$: It generates a random number, $k \in \mathbb{Z}_n$, computes $R = kG$, and converts the field element, R, to pre-signature Π, which is an octet string, with a function, F_Π, i.e., $\Pi = F_\Pi(R)$. The message m is converted to an octet string, d, with a function, F_d, i.e., $d = F_d(m)$. A part of signature, r, is computed from d, Π with a function, F_r, i.e., $r = F_r(d, \Pi)$, and the other part of signature, s, is computed from r, k, x (and m_{clr} if necessary) with a function, F_s, i.e., $s = F_s(r, k, x, m_{\text{clr}})$. Note that F_r and F_s use intermediate integer values, ρ and t, to compute r and s, respectively. If some conditions for ρ, s, or t hold, it recomputes a signature with newly generated k. For the precise recomputing condition in each scheme, consult Section 3.

$\mathcal{S}_V(\mathit{params}, PK, m_{\text{clr}}, \sigma)$: It checks that r' is an octet string with the proper length and that $s' \in \mathbb{Z}_n^*$ hold, where $\sigma = (r', s')$. If not, it outputs `reject`. It computes (a', b') from (r', s') with functions, F'_a and F'_b, i.e.,

$a' = F'_a(r', s')$ and $b' = F'_b(r', s')$. The pre-signature, Π', is computed from a', b', G, and Y with a function, F'_Π, i.e., $\Pi' = F'_\Pi(a', b', G, Y)$. Note that F'_Π uses an intermediate value, R', to compute Π' as $R' = a'G + b'Y$. It computes an octet string, d', from r', Π' with a function, F'_d, i.e., $d' = F'_d(r', \Pi')$, and then, regards d' as m'_{rec}.

After some redundancy check of m'_{rec} (with m'_{clr} if necessary), it outputs the recovered message m' $(= m'_{\mathrm{rec}} || m_{\mathrm{clr}})$ if it passed the check. Otherwise, it outputs `reject`. For the precise rejecting condition in each scheme, consult Section 3.

Note that NR follows this general framework when the additive group is changed to a multiplicative group:

- F_Π is a function which converts a field element to an octet string.
- $d = m_{\mathrm{rec}}$ in F_d.
- $F_r(d, \Pi)$ outputs r, where d and Π are c onverted to integers, δ and π, respectively, $\rho = (\delta + \pi) \bmod n$, and ρ is converted to an octet string, r.
- $F_s(r, k, x, m_{\mathrm{clr}})$ outputs $s = (k - x\rho) \bmod n$, where r is converted to an integer, ρ.
- $a' = s$ in F'_a.
- $F'_b(r, s)$ outputs ρ, where r is converted to an integer, ρ.
- $F'_\Pi(a', b', G, Y)$ outputs Π, where $R' = G^{a'} Y^{b'}$ and R' is converted to a field element, Π.
- $F'_d(r', \Pi')$ outputs m'_{rec}, where r' and Π' are converted to integers, ρ' and π', respectively, $\delta' = (\rho' - \pi') \bmod n$, and δ' is converted to an octet string, m'_{rec}.

4.2.2. GWKS attack on the General Framework.

We describe a gwksa adversary, $\mathcal{A}$, who obtains only single pair, (m, σ), and performs the GWKS attack on the general framework, where $\sigma = (r, s)$:

(1) $\mathcal{A}$ computes (a', b') from (r', s') with the functions, F'_a and F'_b, i.e., $a' = F'_a(r', s')$ and $b' = F'_b(r', s')$.

(2) $\mathcal{A}$ computes the pre-signature, Π', from a', b', G, and Y with the function, F'_Π, i.e., $\Pi' = F'_\Pi(a', b', G, Y)$.

(3) $\mathcal{A}$ generates a random number, $\tilde{x} \in \mathbb{Z}_n^*$.

(4) $\mathcal{A}$ computes $\tilde{G} = \frac{1}{a' + \tilde{x}b'}\tilde{R}$, where $\tilde{R}$ is the intermediate value used in F'_Π and is given as $\tilde{R} = a'G + b'Y$.

(5) $\mathcal{A}$ computes $\tilde{Y} = \tilde{x}\tilde{G}$.

(6) $\mathcal{A}$ outputs $\tilde{G}$, $\tilde{Y}$, $\tilde{x}$, $\tilde{m}$ $(= m)$, and $\tilde{\sigma}$ $(= \sigma)$.

We show that $\mathcal{S}_V$ outputs m ($m = m_{\mathrm{rec}}||m_{\mathrm{clr}}$) when the original signature, σ ($= (r, s)$) and the clear part of the original message, m_{clr}, are given to $\mathcal{S}_V$ with the substituted keys, $(\tilde{G}, \tilde{Y})$, as follows: Note that $G \neq \tilde{G}$, $Y \neq \tilde{Y}$, $m_{\mathrm{clr}} = \tilde{m}_{\mathrm{clr}}$, and $\sigma = \tilde{\sigma}$. $\mathcal{S}_V$ computes $(\tilde{a}', \tilde{b}')$ from (r, s) with functions, F'_a and F'_b. The pre-signature, $\tilde{\Pi}'$, is computed from $\tilde{a}'$, $\tilde{b}'$, $\tilde{G}$, and $\tilde{Y}$ with a function, F'_{Π}, and then, an octet string, $\tilde{d}'$, is computed from r, $\tilde{\Pi}'$ with a function, F_d. We have $\tilde{R}' = \tilde{a}'\tilde{G} + \tilde{b}'\tilde{Y} = \tilde{a}'\left(\frac{1}{\tilde{a}'+\tilde{x}\tilde{b}'}\tilde{R}\right) + \tilde{b}'\left(\tilde{x}\tilde{G}\right) = \tilde{a}'\left(\frac{1}{\tilde{a}'+\tilde{x}\tilde{b}'}\tilde{R}\right) + \tilde{b}'\left(\frac{\tilde{x}}{\tilde{a}'+\tilde{x}\tilde{b}'}\tilde{R}\right) = (\tilde{a}' + \tilde{x}\tilde{b}')\left(\frac{1}{\tilde{a}'+\tilde{x}\tilde{b}'}\tilde{R}\right) = \tilde{R} = a'G + b'Y$. Thus, $m'_{\mathrm{rec}} = \tilde{m}_{\mathrm{rec}}$ holds because $\tilde{R}'$ and $\tilde{R}$ produce the same pre-signature, $\tilde{\Pi}'$ ($= \Pi'$), and $\tilde{m}'_{\mathrm{rec}}$ is recovered from Π' and r' in the same way with m'_{rec} using the function, F_d.

Therefore, $\mathcal{A}$ can succeed in the GWKS attack on the general framework.

In the following, we consider the GWKS security of ECNR, ECMR, ECAO, and ECPV on the general framework:

In the Case of ECNR: The GWKS attack on the general framework with $a = r$, $b = s$ shows that ECNR is vulnerable.

In the Case of ECMR: The GWKS attack on the general framework with $a = \frac{1+r+s}{r}$, $b = \frac{s}{r}$ shows that ECMR is vulnerable.

In the Case of ECAO: The GWKS attack on the general framework with $a = s$, $b = \mathrm{MGF}(r||m_{\mathrm{clr}})$, where MGF is a mask generation function shows that ECAO is vulnerable.

In the Case of ECPV: The GWKS attack on the general framework with $a = s$, $b = \mathrm{MGF}(r||m_{\mathrm{clr}})$, where MGF is a mask generation function shows that ECPV is vulnerable.

We summarize the above considerations in TABLE 1.

TABLE 1. Parameters in unified representation.

scheme	a	b
ECNR	r	s
ECMR	$\frac{1+r+s}{r}$	$\frac{s}{r}$
ECAO	s	$\mathrm{MGF}(r\|\|m_{\mathrm{clr}})$
ECPV	s	$\mathrm{MGF}(r\|\|m_{\mathrm{clr}})$

Concludingly, ECNR, ECMR, ECAO, and ECPV adopted in ISO/IEC 9796--3:2006 also are vulnerable to the GWKS attack.

4.3. Case of ECKNR

ECKNR does not follow the general framework because the additional parameter, z, is used in $\mathcal{S}$ and $\mathcal{S}_V$. We evaluate this effect of z in detail.

Consider an adversary, $\mathcal{A}$, who engages the GWKS attack on ECKNR in similar to the general framework. $\mathcal{A}$ computes $\tilde{R}'$ as $\tilde{R}' = s'G + \rho' Y$, and then, generates substituted keys, $\tilde{G}$, $\tilde{Y}$ as $\tilde{G} = \frac{1}{s' + \tilde{x}\rho'}\tilde{R}'$, $\tilde{Y} = \tilde{x}\tilde{R}'$, where $\tilde{x} \in \mathbb{Z}_n^*$. However, in the signature verification under the substituted keys, $\tilde{d}'$ is generated with the certificate data of the substituted public-key as $\tilde{d}' = r' \oplus \Pi' \oplus \mathrm{MGF}(\tilde{z} || m'_{\mathrm{clr}})$. Thus, it is expected that $\tilde{d}' \neq d'$ holds with high probability when MGF has a collision resistance property. The different $\tilde{d}'$ produces different $\tilde{m}'_{\mathrm{rec}}$ and then, $\tilde{m}'_{\mathrm{rec}}$ might not pass the redundancy check with high probability. Therefore, this naive GWKS attack seem not applicable to ECKNR.

4.4. Summary of Vulnerability

We summarize the security discussion on the MRS schemes.

ISO/IEC 9796–3:2006 adopts six MRSs, NR, ECNR, ECMR, ECAO, ECPV, and ECKNR. We show that five schemes, NR, ECNR, ECMR, ECAO, and ECPV, are vulnerable to the GWKS attack.

We could not show that the GWKS attacks are possible to ECKNR however ECKNR seems secure against the GWKS attacks as it uses the certificate data of the public-key, i.e., z. We stress that it does not mean that ECKNR has provable security against the GWKS attacks.

Table 2. Vulnerability on GWKS attacks.

scheme	GKS attacks
NR	vulnerable
ECNR	vulnerable
ECMR	vulnerable
ECAO	vulnerable
ECPV	vulnerable
ECKNR	unknown[2]

Table 2 shows vulnerability of the MRS schemes against the GWKS attacks.

[2]It is expected to be invulnerable.

5. Conclusions

We formally defined the GKS attacks on MRSs, and stated relations among the KS attacks and the GKS attacks including their strong and weak variants.

We examined vulnerability of the MRS schemes adopted in ISO/IEC 9796-3:2006 against the GWKS attack using an unified representation of the schemes. Consequently, we showed that five among six schemes are vulnerable to the attack.

REFERENCES

[1] ABE, M.—OKAMOTO, T.: *A signature scheme with message recovery as secure as discrete logarithm,* In: (K.-Y. Lam et al., eds.) Advances in Cryptology — ASIACRYPT '99, International Conference on the Theory and Applications of Cryptology and Information Security, Singapore, 1999, Lecture Notes in Comput. Sci. Vol. 1716, Springer-Verlag, Heidelberg (1999), pp. 378–389.

[2] BELLARE, M.—ROGAWAY, P.: *Optimal asymmetric encryption.* In: (A. De Santis, ed.) Advances in Cryptology — EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, 1994, Lecture Notes in Comput. Sci. Vol. 950, Springer-Verlag, Heidelberg, (1995), pp. 92–111.

[3] BLAKE-WILSON, S.— MENEZES, A.: *Unknown key-share attacks on the station-to-station (STS) protocol.* In: (H. Imai, Y. Zheng, eds.) Public Key Cryptography, Second International Workshop on Practice and Theory in Public Key Cryptography, PKC '99, Kamakura, Japan, 1999, Lecture Notes in Comput. Sci. Vol. 1560, Springer-Verlag, Heidelberg, (1999), pp. 154–170.

[4] BOHLI, J.-M.—RÖHRICH, S.—STEINWANDT, R.: *Key substitution attacks revisited: Taking into account malicious signers.* Int. J. Inform. Sec. **5** (2006), no. 1, 30–36.

[5] CHAUM, D.—EVERTSE, J.-H.—VAN DE GRAAF, J.—PERALTA, R.: *Demonstrating possession of a discrete logarithm without revealing it.* In: (A. M. Odlyzko, ed.) Advances in Cryptology — CRYPTO '86, Santa Barbara, California, USA, 1986, Lecture Notes in Comput. Sci. Vol. 263, Springer-Verlag, Heidelberg, (1986), pp. 200–212.

[6] ISO/IEC 14888–2:2008: Information technology — Security techniques — Digital signatures with appendix — Part 2: Integer factorization based mechanisms, 2008.

[7] ISO/IEC 1st CD 14888–3: Information technology — Security techniques — Digital signatures with appendix — Part 3: Discrete logarithm based mechanisms, 2014.

[8] ISO/IEC 14888–3:2006: Information technology — Security techniques — Digital signatures with appendix — Part 3: Discrete logarithm based mechanisms, 2006.

[9] ISO/IEC 9796–2:2010: Information technology — Security techniques — Digital signature schemes giving message recovery — Part 2: Integer factorization based mechanisms, 2010.

[10] ISO/IEC 9796–3:2006: Information technology — Security techniques — Digital signature schemes giving message recovery — Part 3: Discrete logarithm based mechanisms, 2006.

[11] GOLDWASSER, S.—MICALI, S.—RACKOFF, C.: *The knowledge complexity of interactive proof systems.* SIAM J. Comput. **18** (1989), no. 1, 186–208.

[12] MENEZES, A.—SMART, N. P.: *Security of signature schemes in a multi-user setting.* Designs, Codes and Cryptography **33** (2004), no. 3, 261–274.

[13] MIYAJI, A.: *Another countermeasure to forgeries over message recovery signature.* IEICE Transactions on Fundamentals of Electronics, Communications **E80-A** (1997), no. 11, 2192–2200.

[14] NYBERG, K.—RUEPPEL, R. A.: *Message recovery for signature schemes based on the discrete logarithm problem*, Designs, Codes and Cryptography **7** (1996), no. 1–2, 61–81.

[15] PINTSOV, L.—VANSTONE, S.: *Postal revenue collection in the digital age.* In: (Y. Frankel, ed.) Financial Cryptography, 4th International Conference, FC 2000 Anguilla, British West Indies, 2000, Lecture Notes in Comput. Sci. Vol. 1962, Springer-Verlag, Heidelberg, (2000), pp. 105–120.

[16] VAUDENAY, S.: *The security of DSA and ECDSA.* In: (Y. Desmedt, ed.) Public Key Cryptography — PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA , Lecture Notes in Comput. Sci. Vol. 2567, Springer--Verlag, Heidelberg, (2002), pp. 309–323.

[17] VAUDENAY, S.: *Digital signature schemes with domain parameters: Yet another parameter issue in ECDSA.* In: (H. Wang et al., eds.) Information Security and Privacy: 9th Australasian Conference, ACISP 2004, Sydney, Australia, 2004, Lecture Notes in Comput. Sci. Vol. 3108, Springer-Verlag, Heidelberg, (2004), pp. 188–199.

[18] YUM, D. H.—SIM, S. G.—LEE, P. J.: *New signature schemes giving message recovery based on EC-KCDSA.* In: Proceedings of the 12th Conference on Information Security and Cryptology (CISC), (2002), pp. 595–597

[19] ZHANG, Z.—YANG, K.—ZHANG, J.—CHEN, C.: *Security of the SM2 signature scheme against generalized key substitution attacks.* In: (L. Chen, S. Matsuo, eds.) Security Standardisation Research — Second International Conference, SSR 2015, Tokyo, Japan, 2015, Lecture Notes in Comput. Sci. Vol. 9497, Springer-Verlag, Heidelberg, (2015), pp. 140–153.

Appendix A. Relations among KS attacks on MRSs

We summarize relations among four security notions: the WKSA security, the SKSA security, the GWKSA security, and the GSKSA security.

From the definitions, the following proposition and theorem trivially hold.

Proposition 1. *If a MRS scheme is* SKSA *(resp.* GSKSA*) secure, then the scheme is* WKSA *(resp.* GWKSA*) secure.*

Proof. To prove this, we construct a sksa adversary from a wksa adversary.

After the wksa adversary outputs $(\widetilde{PK}, \widetilde{SK})$, the sksa adversary outputs $\widetilde{PK}$. This construction implies that a scheme is not SKSA secure when it is not WKSA secure, and that $\mathbf{Adv}^{\mathsf{sksa}}_{\mathcal{MRS},\mathcal{A}}(\kappa) = \mathbf{Adv}^{\mathsf{wksa}}_{\mathcal{MRS},\mathcal{A}}(\kappa)$ holds. Thus, the case of the key substitution attacks in the proposition is proved.

The case of the generalized attacks can be proved in the similar way. We have $\mathbf{Adv}^{\mathsf{gsksa}}_{\mathcal{MRS},\mathcal{A}}(\kappa) = \mathbf{Adv}^{\mathsf{gwksa}}_{\mathcal{MRS},\mathcal{A}}(\kappa)$.

Both prove the proposition. □

Theorem 1. *If a MRS scheme based on the discrete logarithm problem is* GWKSA *(resp.* GSKSA*) secure, then the scheme is* WKSA *(resp.* SKSA*) secure.*

Proof. To prove this, we construct a gsksa adversary from a sksa adversary.

After the sksa adversary outputs $\widetilde{PK}$, the gsksa adversary outputs $(g, \widetilde{PK})$, where g is the generator in the domain parameter. This construction implies that a scheme is not GSKSA secure when it is not SKSA secure, and that $\mathbf{Adv}^{\mathsf{gsksa}}_{\mathcal{MRS},\mathcal{A}}(\kappa) = \mathbf{Adv}^{\mathsf{sksa}}_{\mathcal{MRS},\mathcal{A}}(\kappa)$ holds. Thus, the case of the weak security in the proposition is proved.

The case of the strong security can be proved in the similar way. We have $\mathbf{Adv}^{\mathsf{gwksa}}_{\mathcal{MRS},\mathcal{A}}(\kappa) = \mathbf{Adv}^{\mathsf{wksa}}_{\mathcal{MRS},\mathcal{A}}(\kappa)$.

Both prove the theorem. □

Here, we introduce a notation, the A security $\Leftarrow$ the B security. This denotes that a scheme is A secure when it is B secure. We may refer to this as the B security *implies* the A security. Thus, the A security $\Leftarrow$ the B security means that the B security is stronger than the A security.

Proposition 1 states that the SKSA security implies the WKSA security and that the GSKSA security implies the GWKSA security.

In signature schemes based on the discrete logarithm problem, **Theorem 1** shows that the gsksa security implies the sksa security and that the gwksa security implies the wksa security, also. We summarize those relations in Figure 1.

$$\begin{tikzcd}
\text{GWKSA security} \arrow[d, Rightarrow] & \text{GSKSA security} \arrow[l, Rightarrow] \arrow[d, Rightarrow] \\
\text{WKSA security} & \text{SKSA security} \arrow[l, Rightarrow]
\end{tikzcd}$$

Figure 1. Relations among KS attacks on MRS under discrete logarithm assumption

Received 29. 9. 2016

Department of Information Systems Creation
Faculty of Engineering
Kanagawa University
3–27–1 Rokkakubashi, Kanagawa,
221–8686 Yokohama
JAPAN
E-mail: fujioka@kanagawa-u.ac.jp