Divertible Protocols and Atomic Proxy Cryptography

Matt Blaze   Gerrit Bleumer   Martin Strauss

AT&T Labs – Research
Florham Park, NJ 07932 USA
{mab,bleumer,mstrauss}@research.att.com

Abstract. First, we introduce the notion of divertibility as a protocol property as opposed to the existing notion as a language property (see Okamoto, Ohta [OO90]). We give a definition of protocol divertibility that applies to arbitrary 2-party protocols and is compatible with Okamoto and Ohta's definition in the case of interactive zero-knowledge proofs. Other important examples falling under the new definition are blind signature protocols. We propose a sufficiency criterion for divertibility that is satisfied by many existing protocols and which, surprisingly, generalizes to cover several protocols not normally associated with divertibility (e.g., Diffie-Hellman key exchange). Next, we introduce *atomic proxy cryptography,* in which an *atomic proxy function,* in conjunction with a public *proxy key,* converts ciphertexts (messages or signatures) for one key into ciphertexts for another. Proxy keys, once generated, may be made public and proxy functions applied in untrusted environments. We present atomic proxy functions for discrete-log-based encryption, identification, and signature schemes. It is not clear whether atomic proxy functions exist in general for all public-key cryptosystems. Finally, we discuss the relationship between divertibility and proxy cryptography.

1 Introduction

This paper investigates two general ways in which an intermediary sitting between the participants of a 2-party protocol might transform the communication messages without "destroying" the protocol. First, we consider *protocol divertibility*, in which the (honest) intermediary, called a *warden*, randomizes all messages so that the intended underlying protocol succeeds, but information contained in subtle deviations from the protocol (for example, information coded into the values of supposedly random challenges) will be obliterated by the warden's transformation. Next, we introduce *atomic proxy cryptography,* in which two parties publish a *proxy key* that allows an untrusted intermediary to convert ciphertexts encrypted for the first party directly into ciphertexts that can be decrypted by the second. The intermediary learns neither cleartext nor secret keys.

Our paper is organized as follows. In Section 2 we discuss divertible protocols. In Section 2.1 we define protocol divertibility. We propose a slightly stricter definition than the original one by Okamoto and Ohta [OO90]. In Section 2.2, we present a sufficiency criterion for divertibility. Its usefulness is demonstrated by many examples of known diverted protocols from the literature. Also many known blind signature protocols can be interpreted as diverted proofs of knowledge and in this form they satisfy our criterion (see [Bleu97]). In Section 3, we introduce atomic proxy cryptography and propose a taxonomy for proxy schemes. In Sections 3.1 to 3.3 we give proxy schemes for encryption, identification, and signature. In Section 4, we discuss the deeper relationship between protocol divertibility and proxy cryptography.

2 Divertible Protocols

The idea of divertibility entered the cryptographic literature during the mid 80's with applications to identification protocols. The basic observation was that some 2-party identification protocols could be extended by placing an intermediary—called a warden for historical reasons [Sim84]—between the prover and verifier so that, even if both parties conspire, they cannot distinguish talking to each other through the warden from talking directly to a hypothetical honest verifier and honest prover, respectively. Since identification protocols were developed in close relation to interactive zero-knowledge proofs (ZKP), Okamoto and Ohta [OO90] (and later Desmedt and Burmester [BD91] and Ihto et al [ISS91]) established the notion of divertibility as a *language property*, i.e., a language is considered divertible if it can be recognized by a diverted interactive zero-knowledge proof system. In this paper, we establish divertibility as a *2-party protocol property*, which is orthogonal to zero knowledge or any other particular protocol property.

2.1 Definitions

In order to deal with protocols of more than two parties, we generalize the notion of *interactive Turing machine* (ITM) by Goldwasser et al [GMR89]. Then we define connections of ITMs and finally give the definition of protocol divertibility.

Definition 1 ((m, n)-Interactive Turing Machine).
An (m, n)-*Interactive Turing Machine* ((m, n)-*ITM*) is a Turing machine with $m \in \mathbb{N}$ read-only *input tapes*, m write-only *output tapes*, m read-only *random tapes*, a *work tape*, a read-only *auxiliary tape*, and $n \in \mathbb{N}_0$ pairs of *communication tapes*. Each pair consists of one read-only and one write-only tape that serves for reading in-messages from or writing out-messages to another ITM. (The purpose of allowing $n = 0$ will become clear below.) The random tapes each contain an infinite stream of bits chosen uniformly at random. Read-only tapes are readable only from left to right. If the string to the right of a read-only head is empty, then we say the tape is *empty*.

Associated to an ITM is a *security parameter* $k \in \mathbb{N}$, a family $D = \{D_\pi\}_\pi$ of tuples of domains, a probabilistic *picking algorithm* $pick(k)$ and an encoding scheme S. Each member

$$D_\pi = (In_\pi^{(1)}, \ldots, In_\pi^{(m)}, Out_\pi^{(1)}, \ldots, Out_\pi^{(m)}, \Omega_\pi^{(1)}, \ldots, \Omega_\pi^{(m)}, \\ (IM_\pi^{(1)}, OM_\pi^{(1)}), \ldots, (IM_\pi^{(n)}, OM_\pi^{(n)}))$$

of D contains one input (output, choice, in-message, out-message) domain for each of the m input (output, random) tapes and n (read-only, write-only) communication tapes. The algorithm $pick(k)$ on input some security parameter k outputs a family index π. Finally, there is a polynomial $P(k)$ so that for each π chosen by $pick(k)$, S encodes all elements of all domains in D_π as bitstrings of length $P(k)$.

ITMs proceed in rounds. During each round, an ITM first reads the messages from all its read-only communication tapes, then performs some computations and finally writes a message to each of its write-only communication tapes. It may write an empty string—denoted ε. If, at the beginning of a round, an ITM finds all its input tapes and all its read-only communication tapes empty, then it performs a last computation, writes empty strings to all its write-only communication tapes, writes results to all its output tapes, and then stops. The overall number of reading, writing and computation steps during an execution of an ITM is bound by a polynomial in the security parameter k.

An (m, n)-ITM is an *m-party protocol* if $n = 0$, and linear if $n \leq 2$. The *native functions* of an ITM A are defined as the family

$$nativ_\pi : \prod_{i=1}^{m} \Omega_{\pi,i} \times \prod_{i=1}^{m} In_{\pi,i} \times \prod_{j=1}^{n} IM_{\pi,j} \rightarrow \prod_{j=1}^{n} OM_{\pi,j}$$

of functions that, on input (rnd, in, im), return the respective out-messages that A would write to its write-only communication tapes would it read this data from its random, input and read-only communication tapes.

Let A be an (m_A, n)-ITM and B be an (m_B, n)-ITM, which together make up a protocol $P = \langle A, B \rangle$. Let $m^* \leq \min(m_A, m_B)$ be the number of pairs of communication tapes shared by A and B. Then the *view* of A on B on respective inputs, denoted as,

$$view_B^{(A)} P([in_{A,1}, \ldots, in_{A,m_A}]^A, [in_{B,1}, \ldots, in_{B,m_B}]^B) \ ,$$

is defined as everything that A sees from B, i.e., the probability distribution of all m^*-tuples of pairs of in-messages sent by A to B and out-messages returned from B to A, where the probabilities are taken over the choices of the viewer A.[1]

For m-party protocols P, we adopt the following interface notation:

$$(out_1, \ldots, out_m) \leftarrow P(in_1, \ldots, in_m) \ ,$$

where the left arrow indicates a probabilistic assignment. If the inputs or outputs consist of several components, we delimit them by square brackets.

[1] This is a generalization of the definition given by Goldwasser, Micali and Rackoff [GMR89].

Definition 2 (Connections of ITMs).
Let A be an (m_A, n_A)-ITM and B be an (m_B, n_B)-ITM with equal picking algorithm *pick*. Then a connection $C = \langle A, B\rangle$ is any ITM consisting of A and B sharing $c \leq \min\{n_A, n_B\}$ pairs of their communication tapes. The picking algorithm of C is *pick*, and the domains of C are defined as the cartesian products of the respective domains of A and B. $\diamond$

Obviously, the linear connection operator $\langle\bullet,\bullet\rangle$ is associative and we can therefore omit brackets in the usual way:

$$\langle A, B, C\rangle \overset{\text{def}}{=} \langle\langle A, B\rangle, C\rangle = \langle A, \langle B, C\rangle\rangle \ .$$

All connections we consider in the following are linear and have a small constant number of rounds.

Definition 3 (Divertibility of Protocols).
Let $P = \langle A, B\rangle$ be a two-party protocol with interface $P([y, x_A]^A, [y, x_B]^B)$ and input domains $In_\pi = (Y_\pi \times X_{A,\pi}) \times (Y_\pi \times X_{B,\pi})$. Common inputs y are taken from Y_π, whereas private inputs x_A, x_B are taken from $X_{A,\pi}$ and $X_{B,\pi}$, respectively. The product domain of private inputs is denoted $X_\pi = X_{A,\pi} \times X_{B,\pi}$. Furthermore, let $R = \{R_\pi\}_\pi$ be a family of relations $R_\pi \subseteq Y_\pi \times X_\pi$.

The protocol P is called *perfectly (computationally) divertible* over R iff a (1,2)-ITM W exists such that the following properties hold:

Extensibility: For all indices π, all common and private inputs $(y, x_A, x_B) \in R_\pi$, the ensembles of views of B on W and on A, i.e.,

$$view_W^{(B)}\langle A,\, W, B\rangle([y, x_A]^A, [y]^W, [y, x_B]^B), \text{ and}$$
$$view_A^{(B)}\langle A, B\rangle([y, x_A]^A, [y, x_B]^B)$$

as well as the views of A on W and on B, i.e.,

$$view_W^{(A)}\langle A,\, W, B\rangle([y, x_A]^A, [y]^W, [y, x_B]^B), \text{ and}$$
$$view_B^{(A)}\langle A, B\rangle([y, x_A]^A, [y, x_B]^B)$$

are equal (polynomially indistinguishable).

Perfect (Computational) Indistinguishability: For all polynomial-time actively adversary ITMs $\tilde{A}$, $\tilde{B}$, for all indices π, all common and private inputs $(y, x_A, x_B) \in R_\pi$ and all polynomial size strings q representing shared a priori knowledge of $\tilde{A}$ and $\tilde{B}$, the ensembles of simultaneous views of $\tilde{A}$ and $\tilde{B}$ upon W and of their views upon honest B and A, i.e.,

$$view_W^{(\tilde{A},\tilde{B})}\langle\tilde{A},\, W, \tilde{B}\rangle([y, x_A, q]^{\tilde{A}}, [y]^W, [y, x_B, q]^{\tilde{B}}) \quad \text{and}$$
$$(view_B^{(\tilde{A})}\langle\tilde{A}, B\rangle([y, x_A, q]^{\tilde{A}}, [y, x_B]^B), view_A^{(\tilde{B})}\langle A, \tilde{B}\rangle([y, x_A]^A, [y, x_B, q]^{\tilde{B}}))$$

are equal (polynomially indistinguishable).[2] [3]

An ITM W that satisfies extensibility and perfect (computational) indistinguishability is said to *perfectly (computationally) divert* protocol P over R. $\diamond$

Divertibility as defined by Okamoto, Ohta [OO90] and almost equivalently by Itoh et al [ISS91] has been introduced as a *language property.* A language L is considered divertible, if there exists a diverted zero knowledge proof system for proving membership in L. In contrast, we define divertibility as a *2-party protocol property.* The main difference between the two definitions is that we ask for a concrete protocol P to be divertible, whereas they ask for existence of a divertible protocol meeting a certain specification S (namely to be a zero-knowledge proof). Consequently, Definition 3 (extensibility) relates the two interfaces of the diverted protocol P' to the interface of the given protocol P, where their definition relates them to S. Another difference is, that we suggest a stronger definition than Okamoto and Ohta's. We require Indistinguishability even for two attackers $\tilde{A}$ and $\tilde{B}$ who *know of each other* (a-priori common knowledge q) and who therefore know which of their views result from the same diverted protocol instance. We discuss this further in Section 2.4.

An immediate consequence of the definition is that if a protocol P is divertible, then we can insert second and third wardens and we, again, obtain a diverted protocol.

2.2 Main Divertibility Result

Theorem 4 (Criterion for Perfect Divertibility).
Let $P = \langle A, B\rangle$ be a two-party protocol with interface $P([y, x_A]^A, [y, x_B]^B)$. Let the input domains be $(Y_\pi \times X_{A,\pi}) \times (Y_\pi \times X_{B,\pi})$, the random domains be $\Omega_{A,\pi} \times \Omega_{B,\pi}$, the out-message domains be $OM_{A,\pi} \times OM_{B,\pi}$, and let the native functions of A and B be

$$\begin{aligned} nativ_{A,\pi} &: \Omega_{A,\pi} \times Y_\pi \times X_{A,\pi} \times OM_{B,\pi} \rightarrow OM_{A,\pi} \quad , \\ nativ_{B,\pi} &: \Omega_{B,\pi} \times Y_\pi \times X_{B,\pi} \times OM_{A,\pi} \rightarrow OM_{B,\pi} \quad . \end{aligned}$$

Furthermore, let $R = \{R_\pi\}_\pi$ be a family of relations $R_\pi \subseteq Y_\pi \times (X_{A,\pi} \times X_{B,\pi})$, which capture the correspondence between the private and the public inputs.

Then P is perfectly divertible over R if only there exist:

(i) a family $(\Omega_\pi, \odot, 1)$ of (not necessarily commutative) groups, and

[2] By $view_B^{(A)} P$, we denote the *view* of A on B in a protocol P. This notion as well as that of *polynomial indistinguishability* of families of random variables is defined, e.g., by Goldwasser, Micali and Rackoff [GMR89].

[3] Equality (polynomial indistinguishability) is required only for the views on *complete* runs of the diverted protocol, i.e., runs that the warden has not aborted, for example, because he has detected either $\tilde{A}$ or $\tilde{B}$ cheating.

(ii) three families of functions

$$\begin{aligned} \mathit{base}_\pi &: Y_\pi \times X_{A,\pi} \times X_{B,\pi} \to OM_{A,\pi} \times OM_{B,\pi} \ , \\ \mathit{join}_\pi &: \Omega_{A,\pi} \times \Omega_{B,\pi} \times Y_\pi \times X_{A,\pi} \times X_{B,\pi} \to \Omega_\pi \ , \\ \mathit{divrt}_\pi &: \Omega_\pi \times Y_\pi \times OM_{A,\pi} \times OM_{B,\pi} \to OM_{A,\pi} \times OM_{B,\pi} \ , \end{aligned}$$

with the following properties:
Function $\mathit{divrt}(\omega, y, o_A, o_B)$ is defined only for (o_A, o_B) that live in the respective image $OM_{\pi,y}$ of native_A and native_B, i.e.,

$$OM_{\pi,y} = \mathit{native}_A(\Omega_A, y, x_A, o_B) \times \mathit{native}_B(\Omega_B, y, x_B, o_A) \ ,$$

where $(y, x_A, x_B) \in R_\pi$.[4]
Second, for each fixed α, β, $y, x_A, x_B \in R_\pi$, the functions,

$$\mathit{join}_\pi(\alpha', \beta, y, x_A, x_B) \quad \textit{and} \quad \mathit{join}_\pi(\alpha, \beta', y, x_A, x_B) \ ,$$

are each bijective on Ω_A and Ω_B, respectively.

(iii) a warden W that on input two in-messages o_A, o_B computes two out-messages o'_A, o'_B such that

$$(o'_A, o_B) = \mathit{divrt}(\omega, y, (o_A, o'_B)) \ .$$

Now, for every π, for all random choices $\alpha \in \Omega_{A,\pi}, \beta \in \Omega_{B,\pi}$, all common and corresponding private inputs $(y, x_A, x_B) \in R_\pi$, and all out-messages $o_A \in OM_{A,\pi}$, $o_B \in OM_{B,\pi}$ the following three conditions must hold:

Decomposition:

$$\begin{aligned} &(\mathit{nativ}_A(\alpha, y, x_A, o_B), \mathit{nativ}_B(\beta, y, x_B, o_A)) \\ &= \mathit{divrt}(\mathit{join}(\alpha, \beta, y, x_A, x_B), y, \mathit{base}(y, x_A, x_B)) \ , \end{aligned}$$

Ground:

$$\mathit{divrt}(1, y, (o_A, o_B)) = (o_A, o_B) \ ,$$

Mixed Associativity:

$$\mathit{divrt}(\omega', y, \mathit{divrt}(\omega, y, (o_A, o_B))) = \mathit{divrt}(\omega \odot \omega', y, (o_A, o_B)) \ .$$

◇

Proof. First observe that if *divrt* satisfies the premises Ground and Mixed Associativity, then it is injective as a function of ω: For all $(o_A, o_B) \in OM_{\pi,y}$, we have:

$$\begin{aligned} (o_A, o_B) = {} & \mathit{divrt}(1, y, (o_A, o_B)) \\ & \mathit{divrt}(\omega \odot \omega^{-1}, y, (o_A, o_B)) \ \text{(for any } \omega) \\ & \mathit{divrt}(\omega^{-1}, y, \mathit{divrt}(\omega, y, (o_A, o_B))) \ . \end{aligned}$$

[4] Note that the input variables o_A and o_B in the definition of $OM_{\pi,y}$ refer to the output of nativ_B and nativ_A, respectively. This recursion is guaranteed to terminate by the following requirement (iii) below.

So, function *divrt* turns out to be bijective on Ω_π for the entire parameter domain $OM_{\pi,y}$. We may thus write: $\mathit{divrt}^{-1}(\omega, \bullet) = \mathit{divrt}(\omega^{-1}, \bullet)$.

In order to infer extensibility and indistinguishability of P, we look separately at the out-messages between $\langle A, W\rangle$ and B and those out-messages between A and $\langle W, B\rangle$. We deal with the former case in detail and argue that the latter case can be handled analogously due to symmetry reasons. Using Decomposition and Mixed Associativity, we rewrite the above mentioned out-messages as follows:

$$\begin{aligned}
&(\mathit{nativ}_{\langle A,W\rangle}(\omega, \alpha, y, x_A, o_B), \mathit{nativ}_B(\beta, y, x_B, o_A)) \\
&= \mathit{divrt}(\omega, y, (\mathit{nativ}_A(\alpha, y, x_A, o'_B), \mathit{nativ}_B(\beta, y, x_B, o'_A))) \\
&= \mathit{divrt}(\omega, y, \mathit{divrt}(\omega', y, \mathit{base}(y, x_A, x_B))), \\
&\qquad \text{where } \omega' = \mathit{join}(\alpha, \beta, y, x_A, x_B) \\
&= \mathit{divrt}(\omega' \odot \omega, y, \mathit{base}(y, x_A, x_B)) \\
&= (\mathit{nativ}_A(\alpha', y, x_A, o_B), \mathit{nativ}_B(\omega, \beta', y, x_B, o_A)) \enspace , \\
&\qquad \text{where } (\alpha', \beta') = \mathit{join}^{-1}(\omega' \odot \omega, y, x_A, x_B) \enspace .
\end{aligned}$$

It then follows from the bijectiveness of *join* and the fact that $\odot$ is a group operation that the probability of each pair of out-messages is the same over Bob's choices β and over β'. Together with the analogous result for out-messages between A and $\langle W, B\rangle$ (this is where invertibility of *divrt* is needed), this settles extensibility.

For perfect indistinguishability, we need to deal with arbitrary attackers $\tilde{A}$, $\tilde{B}$, instead. Assume, these attackers produce their out-messages with a certain distribution D that respects the domain of function *divrt*. Otherwise, *divrt* is undefined and the distribution could be ignored according to indistinguishability. Then by decomposition, we see that this given distribution D can also be achieved by honest Alice and Bob if Bob would chose his β according to some appropriate distribution d. Following the above rewriting, and again taking into account that join is bijective and $\odot$ is a group operation, we conclude, that the distribution of $\omega' \odot \omega$ is d because, by presumption, the warden is honest and therefore ω is uniformly distributed. Hence, the out-messages of $\langle A, W\rangle$ and B are also distributed according to D, if the probabilities are taken over β'. Together with the analogous result for out-messages between A and $\langle W, B\rangle$, this in addition settles perfect indistinguishability and therefore perfect divertibility. □

2.3 New Example of Diverted Protocol

The most prominent examples of diverted protocols in the literature are diverted interactive proofs and blind signatures. Since divertibility has been introduced only in the former context, blind signatures are a good example to illustrate the more general concept of divertibility of protocols as proposed in Definition 3. The practical value of Theorem 4 is demonstrated in [Bleu97] by proving many protocols unconditionally divertible; in particular (i) the diverted ZKP that Okamoto and Ohta used to prove their main theorem [OO90] and (ii) a blind modified ElGamal Signature, which was presented by Horster, Michels and Petersen [HMP95] who built on ideas of Camenisch, Piveteau and Stadler [CPS95].

Here, we consider a new sort of protocol for divertibility, namely key exchange. In Figure 1, we present a diverted Diffie-Hellman key exchange protocol [DH76]. Let p be a k-bit prime ($k \in \mathbb{N}$), q be a large prime divisor of $p-1$ and G_q be the unique (multiplicative) subgroup of order q in $\mathbb{Z}_p^*$. Furthermore, $g \neq 1$ denotes a randomly chosen element of G_q. (The restriction to $g \neq 1$ asserts that g generates G_q). p, q and g are global system parameters and neither Alice nor Bob have private inputs.

$$([a'']^A, [b'']^B) \leftarrow DDH([]^A, []^W, []^B)$$

	Alice		**Warden**		**Bob**
(1)	$\alpha \in_R \mathbb{Z}_q$		$\omega \in_R \mathbb{Z}_q$		$\beta \in_R \mathbb{Z}_q$
(2)	$a \leftarrow g^\alpha$				$b \leftarrow g^\beta$
(3)		$\xrightarrow{a}$	$(a', b') \leftarrow (a^\omega, b^\omega)$	$\xleftarrow{b}$	
(4)	$a'' \leftarrow b'^\alpha$	$\xleftarrow{b'}$		$\xrightarrow{a'}$	$b'' \leftarrow a'^\beta$

Fig. 1. Diverted Diffie-Hellman Key Exchange

Proposition 5. *The warden of protocol DDH computationally diverts the Diffie-Hellman protocol between Alice and Bob over $R = \emptyset$.* ◇

Proof (Sketch). If for given (a, b), an attacker could distinguish valid from invalid diverted out-messages (a', b') with non-negligible probability, i.e., probability $\geq \frac{1}{P(k)}$ for some polynomial P, then he had broken the simultaneous discrete log assumption [CEG88]. □

2.4 Why the Previous Definition is a Little too Weak

The previous definition of divertibility by Okamoto and Ohta [OO90], and by Itoh et al [ISS91] as well, requires that two attackers $\tilde{A}, \tilde{B}$ who on the one hand form a linear 3-party protocol P' with an intermediate warden and on the other hand form 2-party protocols $\langle \tilde{A}, B \rangle$ with an honest B and $\langle A, \tilde{B} \rangle$ with an honest A cannot distinguish their views in $\langle \tilde{A}, B \rangle$ and $\langle A, \tilde{B} \rangle$ from those in *separate* instances of $\langle \tilde{A}, W, \tilde{B} \rangle$. More formally, they require indistinguishability of the two ensembles (protocol inputs exactly as in Definition 3 before):

$$(view_W^{(\tilde{A})} \langle \tilde{A}, W, \tilde{B} \rangle, view_W^{(\tilde{B})} \langle \tilde{A}, W, \tilde{B} \rangle) \qquad (1)$$

$$\text{and} \quad (view_B^{(\tilde{A})} \langle \tilde{A}, B \rangle, view_A^{(\tilde{B})} \langle A, \tilde{B} \rangle). \qquad (2)$$

However, the attacker model that seems to underly the literature on divertibility is stronger than expressed by the above requirement. The attackers $\tilde{A}$ and $\tilde{B}$ are usually assumed to know when they engage in a protocol with the warden and so they know which of their views result from the same protocol instances.

A good example to illustrate this difference is protocol *DDH* in Section 2.3. The two ensembles according to (1) and (2) above are equal and thus protocol DDH would have to be regarded as perfectly diverted. This is counterintuitive because the warden in DDH uses less random coins than Alice and Bob together. On the other hand, according to Definition 3, DDH is only computationally diverted, which is the most we would expect.

3 Atomic Proxy Cryptography

A basic goal of public-key encryption is to allow only the key or keys selected at the time of encryption to decrypt the ciphertext. To change the ciphertext to a different key requires re-encryption of the message with the new key, which implies access to the original cleartext and to a reliable copy of the new encryption key. Intuitively, this seems a fundamental, and quite desirable, property of good cryptography; it should not be possible for an untrusted party to change the key with which a message can be decrypted.

Here, on the other hand, we investigate the possibility of *atomic proxy functions* that convert ciphertext for one key into ciphertext for another without revealing secret decryption keys or cleartext messages. An atomic proxy function allows an untrusted party to convert ciphertext between keys without access to either the original message or to the secret component of the old key or the new key. In proxy cryptography, the holders of public-key pairs A and B create and publish a *proxy key* $\pi_{A \to B}$ such that $D(\mathit{\Pi}(E(m, e_A), \pi_{A \to B}), d_B) = m$, where $E(m, e)$ is the public encryption function of message m under encryption key e, $D(c, d)$ is the decryption function of ciphertext c under decryption key d, $\mathit{\Pi}(c, \pi)$ is the atomic proxy function that converts ciphertext c according to proxy key π, and e_A, e_B, d_A, d_B are the public encryption and secret decryption component keys for key pairs A and B, respectively. The proxy key gives the owner of B the ability to decrypt "on behalf of" A; B can act as A's "proxy." In other words, the $\mathit{\Pi}$ function effectively allows the "atomic" computation of $E(D(c, d_A), e_B)$ without revealing the intermediate result $D(c, d_A)$.

We consider atomic proxy schemes for encryption, identification and signatures. An encryption proxy key $\pi_{A \to B}$ allows B to decrypt messages encrypted for A and an identification or signature proxy key $\pi_{A \to B}$ allows A to identify herself as B or to sign for B (i.e., transforms A's signature into B's signature). Generating encryption proxy key $\pi_{A \to B}$ obviously requires knowledge of at least the secret component of A (otherwise the underlying system is not secure) and similarly generating identification or signature proxy key $\pi_{A \to B}$ requires B's secret, but the proxy key itself, once generated, can be published safely.

Categories of proxy schemes Encryption proxy functions (and similarly but contravariantly, identification or signature proxy functions) can be categorized according to the degree of trust they imply between the two key holders. Clearly, A must (unconditionally) trust B, since the encryption proxy function by definition allows B to decrypt on behalf of A. *Symmetric* proxy functions also imply that B trusts A, e.g., because d_B can be feasibly calculated given the proxy key plus d_A. *Asymmetric* proxy functions do not imply this bilateral trust. (Note that this model implies that proxy cryptography probably makes sense only in the context of public-key cryptosystems. Any secret-key cryptosystem with an asymmetric proxy function could be converted into a public-key system by publishing one key along with a proxy key that converts ciphertext for that key into ciphertext for a second key (which is kept secret.))

We can also categorize the asymmetric proxy schemes that might exist according to the convenience in creating the proxy key. In an *active asymmetric* scheme, B has to cooperate to produce the proxy key $\pi_{A \to B}$ feasibly, although the proxy key (even together with A's secret key) might not compromise B's secret key. In a *passive asymmetric* scheme, on the other hand, A's secret key and B's public key suffice to construct the proxy key. Clearly, any passive asymmetric scheme can be used as an active asymmetric scheme, and any asymmetric scheme can be used as a symmetric scheme.

Finally, we can (informally) distinguish proxy schemes according to the "metadata" they reveal about the identity of the secret-public key-pairs being transformed. *Transparent* proxy keys reveal the original two public keys to a third party. *Translucent* proxy keys allow a third party to verify a guess as to which two keys are involved (given their public keys). *Opaque* proxy keys reveal nothing, even to an adversary who correctly guesses the original public keys (but who does not know the secret keys involved).

Proxy schemes in theory and practice The proxy relationship is necessarily transitive. If there are public proxy keys $\pi_{A \to B}$ and $\pi_{B \to C}$, then anyone can compute a proxy function for $A \to C$. Symmetric proxy schemes further establish equivalence classes of keys where the secret component of any key can be used to decrypt messages for any other key in the same class. Note that creating a single symmetric proxy key between a key in one class and a key in another effectively joins the two classes into one.

The notion of proxy cryptography is a rather natural generalization of public-key cryptography and has some nice theoretical properties. The proxy schemes we consider below have the additional property that anyone can use the proxy key $\pi_{A \to B}$ to transform the public key of A to the public key of B. For such proxy schemes, as we will see in the various examples below, certain aspects of the security of publishing a proxy key actually follow from the fact that anyone, trusted or not, can use a proxy key to transform ciphertext and keys.

For example, suppose random messages m and m' are encrypted with random secret keys a and b as $E(m, a)$, $E(m', b)$. Suppose that knowing the proxy key $\pi_{A \to B}$ enables Eve, who knows neither a nor b, to recover m or m'. Then, ignoring

B altogether and starting with just two (presumably secure) ciphertexts $E(m, a)$ and $E(m', a)$, Eve can pick a random proxy key $r = \pi_{A \to Q}$ for some Q, transform $E(m', a)$ to $E(m', q)$ (where q is the unknown secret key of Q), transform A's public key into Q's public key, and proceed with the hypothesized cryptanalysis. We conclude that if it is safe for A to publish k messages then it is safe for A and B to publish a total of k messages *and* to publish a proxy key, provided only that Eve can successfully *apply* the proxy key to transform ciphertext and public keys.

Because proxy keys are tied to specific key pairs, it is not necessary in many applications to certify or otherwise take special care in distributing them (except to prevent denial-of-service). In particular, it is generally sufficient to rely on the certification and trust established in A (for encryption) or B (for signatures) when using proxy key $\pi_{A \to B}$, since a valid proxy key can by definition only be generated with the cooperation of the owner. Furthermore, the proxy function can be safely applied at any convenient time or place, by the message's sender or receiver, or at any intermediate (and possibly untrusted) point in the network.

Proxy functions potentially also have practical utility for key management in real systems. For example, some pieces of secure hardware (*e.g.,* smartcards) limit the number of secret keys that can be stored in secure memory, while some applications might require the ability to decrypt messages for more keys than the hardware can accommodate. With proxy cryptography, once a new key is created and a corresponding proxy key generated, the secret component of the old (or new) key can be destroyed, with the (public and externally-applied) proxy key maintaining the ability to decrypt for both. In effect, proxy functions allow us to increase the number of public keys without also increasing the number of secret bits or the amount of secret computation. Because proxy functions can be computed anywhere, messaging systems, such as electronic mail, can proxy "forward" messages encrypted with one key to a recipient who holds a different key. Proxy functions make it possible to associate a single key with a network or physical address but still decrypt messages forwarded (and proxied) from other addresses. Finally, proxy functions effectively allow changing or adding a key without obtaining new certificates or altering the distribution channel for the previous public key; this could be useful when it is difficult to distribute or certify new keys (e.g., old keys were published in widely-distributed advertisements or embedded in published software, or the certification authority charges high fees for new certificates).

Security of proxy schemes and ad hoc substitutes If Alice wants Bob to be able to read her mail, instead of issuing a proxy key she might just give Bob her secret key (perhaps, obviating the need to involve Bob, by encrypting it in Bob's public key and publishing it). This would be inferior to using a proxy scheme for several reasons. First, as discussed above, Bob's computing environment may be limited and therefore incapable of automatically processing encrypted secret keys; any new software to decrypt and manage such keys would have to run within the environment trusted by Bob. Proxy processing, on the other hand, can take place entirely outside of Alice's and Bob's trusted environments and without their active involvement. Furthermore, encrypting one's secret key with another's public key is not in general secure. The cryptosystem we present below, a variant of ElGamal [ElG85], is thought to be secure in part because the cryptanalysis problem is random-self-reducible—which allows one to assert mathematically that recovering m from the public information $\langle e_a, E(m, e_a), e_b \rangle$ is hard on average if it is hard at worst. The task of recovering m from $\langle e_a, E(m, e_a), E(d_a, e_b), e_b \rangle$, however, may be considerably easier since $E(d_a, e_b)$, in the context of e_a and e_b, may leak information about d_a—specifically, the new cryptanalysis problem is probably not random-self-reducible and due to the problem's obscurity it is not clear what, if any, mathematical guarantees of security can be given. By contrast, the proxy scheme we give below is just as strong as the underlying cryptosystem.[5]

Related work A natural question to ask is whether there exist atomic proxy functions (and feasible schemes to generate proxy keys) for any public key cryptosystems.

Previous work on delegating the power to decrypt has focused on developing efficient transformations that allow the original recipient to forward *specific ciphertexts* to another recipient. Mambo and Okamoto [MO97] develop this formulation and give efficient transforms (more efficient than decryption and re-encryption) for ElGamal and RSA. Mambo, Usuda and Okamoto [MUO96] apply a similar notion to signature schemes.

While such schemes have value from the standpoint of efficiency, they are not, however, "atomic proxy cryptosystems" by our definition because the transforming function must be kept secret and applied online by the original keyholder on a message-by-message basis (the schemes are not atomic). The security semantics of these systems are essentially the same as a decryption operation followed by a re-encryption operation for the new recipient. Our formulation of proxy cryptography is distinguished from the previous literature by the ability of the keyholder to publish the proxy function and have it applied by untrusted parties without further involvement by the original keyholder.

3.1 Proxy encryption

Although the problem of proxy cryptography seems like a natural extension of public-key cryptography, existing cryptosystems do not lend themselves to obvious proxy functions. RSA [RSA78] with a common modulus is an obvious candidate, but that scheme is known to be insecure [Sim83,DeL84]. Similarly, there

[5] Note that Bob of this example may be a government mandating that Alice provide him with access to her key. It has been argued that such a scheme makes the system as a whole less trustworthy due to the extra engineering effort involved; we argue here that in the case of random-self-reducible cryptosystems such as ElGamal variants, requiring Alice to encrypt her secret key using the government's public key may also weaken the underlying cryptosystem in the precise mathematical sense of spoiling the random-self-reducibility.

do not appear to be obvious proxy functions for many of the previous discrete-log-based cryptosystems. This is not to say, of course, that proxy functions for existing systems do not exist.

We now describe a new secure discrete-log-based public-key cryptosystem that does have a simple proxy function. The scheme is similar in structure to ElGamal encryption [ElG85], but with the parameters used differently and the inverse of the secret used to recover the message.[6] (This approach has merit beyond proxy encryption; [Hug94] proposed a Diffie-Hellman-like key agreement protocol based on the inverse of the secret, which allows a message's sender to determine the key prior to identifying its recipient).

Cryptosystem $\mathcal{X}$ (encryption) Let p be a prime of the form $2q + 1$ for a prime q and let g be a generator in $\mathbb{Z}_p^*$; p and g are global parameters shared by all users. A's secret key a, $0 < a < p - 1$, is selected at random and must be in $\mathbb{Z}_{2q}^*$, *i.e.,* relatively prime to $p - 1$. (A also calculates the inverse a^{-1} mod $2q$). A publishes the public key $g^a \bmod p$. Message encryption requires a unique randomly-selected secret parameter $k \in \mathbb{Z}_{2q}^*$. To encrypt m with A's key, the sender computes and sends two ciphertext values (c_1, c_2):

$$c_1 = mg^k \bmod p$$
$$c_2 = (g^a)^k \bmod p$$

Decryption reverses the process; since

$$c_2^{(a^{-1})} = g^k \pmod{p}$$

it is easy for A (who knows a^{-1}) to calculate g^k and recover m:

$$m = c_1((c_2^{(a^{-1})})^{-1}) \bmod p$$

The efficiency of this scheme is comparable to standard ElGamal encryption.

Symmetric proxy function for $\mathcal{X}$ Observe that the c_1 ciphertext component produced by Cryptosystem $\mathcal{X}$ is independent of the recipient's public key. Recipient A's key is embedded only in the c_2 exponent; it is sufficient for a proxy function to convert ciphertext for A into ciphertext for B to remove A's key a from c_2 and replace it with B's key b. Part of what a proxy function must do, then, is similar to the first step of the decryption function, raising c_2 to a^{-1} to remove a. The proxy function must also contribute a factor of b to the exponent. Clearly, simply rasing c_2 to a^{-1} and then to b would accomplish this, but obviously such a scheme would not qualify as a secure proxy function; anyone who examines the proxy key learns the secret keys for both A and B.

This problem is avoided, of course, by combining the two steps into one. Hence, the proxy key $\pi_{A \to B}$ is $a^{-1}b$ and the proxy function is simply $c_2^{\pi_{A \to B}}$.

[6] David Wagner notes that this proxy scheme can be extended to work with standard ElGamal encryption.

Note that this is a symmetric proxy function; A and B must trust one another bilaterally. B can learn A's secret (by multiplying the proxy key by b^{-1}), and A can similarly discover B's key. Observe that applying the proxy function is more efficient than decryption and re-encryption, in that only one exponentiation is required.

Security of $\mathcal{X}$ First, we show that $\mathcal{X}$ is secure—that cleartext and secret keys cannot be recovered from ciphertext and public keys. Beyond that, we also show that publishing the proxy key compromises neither messages nor secret keys. Since recovering a secret key enables an adversary to recover a message and since cryptanalysis is easier with more information (i.e., a proxy key), it is sufficient to show that no cleartext is recoverable from ciphertext, public keys, and proxy keys. Specifically, we will show that the problem of recovering m from

$$(g^a, g^b, g^c, \ldots, mg^k, g^{ak}, a^{-1}b, a^{-1}c, \ldots).$$

is at least as hard as Diffie-Hellman.

Theorem 6. *Suppose there exists a randomized algorithm f that with probability $\epsilon > 1/|p|^{O(1)}$ succeeds in recovering m from the public information*

$$(g^a, g^b, \ldots, mg^k, g^{ak}, b/a, \ldots)$$

where the probability is taken over f's random choices as well as over m and the parameters a, b, and k. Then, for each $\eta = 2^{-|p|^{O(1)}}$, there exists a randomized polynomial-time algorithm for Diffie-Hellman that succeeds with probability $1-\eta$.

Proof. The proof is found in [BS98].

Similarly one can show that recovering a from $(g^a, g^b, mg^k, g^{ak}, b/a)$ is as hard as the discrete log, so publishing the proxy key does not compromise a—not even to the level of Diffie-Hellman.

3.2 Proxy identification

In this section we describe a discrete-log-based identification scheme. With p, g, a as before, Alice wishes to convince Charlotte that she controls a; Charlotte will verify using public key g^a. As before, the proxy key $\pi_{A \to B}$ will be a/b—it will be safe to publish a/b and Alice and Charlotte can easily use a/b to transform the protocol so Charlotte is convinced that Alice controls b.

Note that in the case of a secure identification proxy key that transforms identification by A into identification by B, it is B whose secret is required to construct the proxy key because identification as B should not be possible without B's cooperation.

Cryptosystem $\mathcal{Y}$ (identification) Let p and g be a prime and a generator in $\mathbb{Z}_p^*$, respectively. Alice picks random $a \in \mathbb{Z}_{2q}^*$ to be her secret key and publishes g^a as her public key. Each round of the identification protocol is as follows:

- Alice picks a random $k \in \mathbb{Z}_{2q}^*$ and sends Charlotte $s_1 = g^k$.
- Charlotte picks a random bit and sends it to Alice.
- Depending on the bit received, Alice sends Charlotte either $s_2 = k$ or $s_2' = k/a$.
- Depending on the bit, Charlotte checks that $(g^a)^{s_2'} = s_1$ or that $g^{s_2} = g^k$.

This round is repeated as desired. As with existing protocols, there may be ways to perform several rounds in parallel for efficiency [FFS88].

Symmetric proxy function for $\mathcal{Y}$ A symmetric proxy key is a/b. Suppose Charlotte wants to run the protocol with g^b instead of g^a. Either Alice or Charlotte or any intermediary can use the proxy key to convert Alice's responses k/a to k/b.

Security of $\mathcal{Y}$

Theorem 7. *Protocol $\mathcal{Y}$, with or without proxy keys published, is a zero knowledge protocol that convinces the verifier that the prover knows the secret key.*

Proof. The proof is found in [BS98].

3.3 Proxy signature

The concept of proxy cryptography also extends to digital signature schemes. A signature proxy function transforms a message signature so that it will verify with a public key other than that of the original signer. In other words, a signature proxy function $\Pi(s, \pi_{A \to B})$ with proxy key $\pi_{A \to B}$ transforms signature s signed by the secret component of key A such that $V(m, \Pi(S(m, A), \pi_{A \to B}), B)$ returns VALID, where $S(m, k)$ is the signature function for message m by key k and $V(m, s, k)$ is the verify function for message m with signature s by key k.

Again, existing digital signature schemes such as RSA [RSA78], DSA [NIS91], or ElGamal [ElG85], etc. do not have obvious proxy functions (which, again, is not to say that such functions do not exist).

As in the case of proxy identification, in order to construct a proxy key that transforms A's signature into B's signature, B's secret must be required to construct the proxy key because signing for B should not be possible without B's cooperation.

Now we will see how to use the proxy identification scheme to construct a proxy signature scheme. We suppose there exists a hash function h whose exact security requirements will be discussed below. The parameters p, g, a, b are as before.

Cryptosystem $\mathcal{Z}$ (signature) To sign a message m, Alice picks $k_1, k_2, \ldots k_\ell$ at random and computes $g^{k_1}, \ldots g^{k_\ell}$. Next Alice computes $h(g^{k_1}, \ldots g^{k_\ell})$ and extracts ℓ pseudorandom bits $\beta_1, \ldots, \beta_\ell$. For each i, depending on the i'th pseudorandom bit, Alice (who knows a) computes $s_{2,i} = (k_i - m\beta_i)/a$; that is, $s_{2,i} = (k_i - m)/a$ or $s_{2,i} = k_i/a$. The signature consists of two components:

$$\begin{aligned} s_1 &= (g^{k_1}, \ldots, g^{k_\ell}) \\ s_2 &= ((k_1 - m\beta_1)/a, \ldots, (k_\ell - m\beta_\ell)/a) \end{aligned}$$

To verify the signature, first the β_i's are recovered using the hash function. The signature is then verified one "round" at a time, where the i'th round is $(g^{k_i}, (k_i - m\beta_i)/a)$. To verify $(g^k, (k - m\beta)/a)$ using public key g^a, the recipient Charlotte raises (g^a) to the power $(k - m\beta)/a$ and checks that it matches $g^k/g^{m\beta}$.

Symmetric proxy function for $\mathcal{Z}$ A symmetric proxy key $\pi_{A\rightarrow B}$ for this signature scheme is a/b. The proxy function Π leaves s_1 alone and maps each component $s_{2,i}$ to $s_{2,i}\pi_{A\rightarrow B}$.

Security of $\mathcal{Z}$ This scheme relies on the existence of a "hash" function h. Specifically (Hash Assumption), we assume there exists a function h such that:

- On random input (g^a, m), it is difficult to generate $\{r_i\}$ and $\{\beta_i\}$ such that
$$h(g^{ar_1 + m\beta_1}, \ldots, g^{ar_\ell + m\beta_\ell}) = \langle \beta_1 \ldots, \beta_\ell \rangle.$$
- More generally, it is difficult to generate such $\{r_i\}$ and $\{\beta_i\}$ on input g^a, m, and samples of signatures on random messages signed with a.

It is not our intention to conjecture about the existence of such functions h. In particular, we do not know the relationship between the hash assumption and assumptions about collision freedom or hardness to invert.[7] We note that this generic transformation of a protocol to a signature scheme has appeared in the literature [FS87].

We now analyze the hash assumption. Note that in order to produce a legitimate signature on m that verifies with g^a, a signer needs to produce $\langle g^{k_i} \rangle$ and $\langle (k_i - m\beta_i)/a \rangle$. Thus, putting $\langle \beta_i \rangle = h(\langle g^{k_i} \rangle)$ and then $\langle r_i \rangle = \langle (k_i - m\beta_i)/a \rangle$, it is straightforward to see that the signer could actually produce r_i's and β_i's of the stated type in the course of producing the signature.

While we do not address the security of h, we can state that issuing proxy keys does not weaken the system.

Theorem 8. *Suppose h satisfies the hash assumption. Then, for most b, it is also hard to produce $\{r_i\}$ and $\{\beta_i\}$ given additional input $a/b, g^b$, and samples of messages signed with b.*

Proof. The proof is found in [BS98].

[7] The hash assumption *does* imply that, on random input g^a, it is hard to find $\langle r_i \rangle$ making all the β_i's zero, i.e., such that $h(g^{ar_1}, \ldots, g^{ar_\ell}) = 0$.

4 Conclusions

Conceptually, divertibility and proxiability of protocols are both defined in terms of an effectiveness property and one or two security properties. The effectiveness property is basically the same in both cases, namely extensibility as in Definition 3. Our more recent work shows that a proxy key can be naturally incorporated into (and makes sense for) divertibility as well. In the case of divertibility, the security requirement is that Alice and Bob cannot communicate any subliminal message through the warden. In the case of proxiability, the security requirement is that the proxy key releases no more information than what either Alice or Bob would release in the original protocol. A complete unifying framework remains as future work.

We have introduced the notion of perfect and computational protocol divertibility, and have given a sufficiency criterion for the former. All existing diverted protocols we have found in the literature turned out to satisfy this criterion. The first example of a diverted key distribution protocol was given. This is also the first computationally divertible protocol we know of.

Intuitively, atomic proxy cryptography is a fairly natural extension of the basic notion of public-key cryptography. It surely seems plausible, given that there exist cryptosystems that can grant the ability to encrypt without granting the ability to decrypt, that there might also exist cryptosystems that can grant the ability to *re-encrypt* without granting the ability to decrypt. However, it is not at all obvious whether there exist atomic proxy schemes in general.

Indeed, while this paper demonstrates that there do exist efficient and secure public-key encryption and signature schemes with symmetric atomic proxy functions, this observation probably raises more new questions than it answers. In particular, do proxy functions exist for public-key cryptosystems based on problems other than discrete-log? (One possibility is that, for some cryptosystems, proxy functions do exist but it is infeasible to find a proxy key.) More importantly, we have yet to discover a secure *asymmetric* proxy function of any kind; asymmetric proxy functions are probably much more useful in practice, since there are likely many situations where trust is only unidirectional. Are there cryptosystems for which asymmetric proxy functions exist?

5 Acknowledgements

We thank Steve Bellovin, Matthew Franklin, Jack Lacy, Dave Maher, Andrew Odlyzko and David Wagner for helpful discussions and comments.

References

[BS98] Matt Blaze, Martin Strauss. Atomic Proxy Cryptography. AT&T Labs-Research TR98.5.1 <http://www.research.att.com/library/trs>

[Bleu97] Gerrit Bleumer. On Protocol Divertibility. AT&T Labs-Research TR97.34.2 <http://www.research.att.com/library/trs>

[BD91] Mike V. D. Burmester, Yvo Desmedt. All languages in NP have divertible zero-knowledge proofs and arguments under cryptographic assumptions. *Eurocrypt '90* LNCS 473, Springer-Verlag 1991, 1–10.

[CEG88] David Chaum, Jan-Hendrik Evertse, Jeroen van de Graaf. An improved protocol for demonstrating possession of discrete logarithms and some generalizations. *Eurocrypt '87.* LNCS 304, Springer-Verlag 1988, 127–141.

[CPS95] Jan L. Camenisch, Jean-Marc Piveteau, Markus A. Stadler. Blind Signatures Based on the Discrete Logarithm Problem. *Eurocrypt '94.* LNCS 950, Springer-Verlag 1995, 428–432.

[DeL84] John M. DeLaurentis. A Further Weakness in the Common Modulus Protocol for the RSA Cryptoalgorithm. *Cryptologia* 8/3 (1984) 253–259.

[DH76] Whitfield Diffie, Martin E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory.* 22/6 (1976) 644–654.

[ElG85] Taher ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory.* 31/4 (1985) 469–472.

[FFS88] Uriel Feige, Amos Fiat, Adi Shamir. Zero-Knowledge Proofs of Identity. *Journal of Cryptology* 1/2 (1988) 77–94.

[FS87] Amos Fiat, Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. *Crypto '86.* LNCS 263, Springer-Verlag 1987, 186–194.

[GMR89] Shafi Goldwasser, Silvio Micali, Charles Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM J. Computing.* 18/1 (1989) 186–207.

[HMP95] Patrick Horster, Markus Michels, Holger Petersen. Meta-Message Recovery and Meta-Blind Signature Schemes Based on the Discrete Logarithm Problem and Their Applications. *Asiacrypt '94.* LNCS 917, Springer-Verlag 1995, 224–237.

[Hug94] Eric Hughes. An encrypted key transmission protocol. *Crypto '94 Rump Session* presentation, August 1994.

[ISS91] Toshija Itoh, Kouichi Sakurai, Hiroki Shizuya. Any Language in IP has a Divertible ZKIP. *AsiaCrypt '91.* Springer-Verlag 1993, 382–396.

[MO97] Masahiro Mambo, Eiji Okamoto. Proxy cryptosystems: delegation of the power to decrypt ciphertexts. *IEICE Trans. Fund. Electronics Communications and Comp Sci.* E80-A/1 (1997) 54–63.

[MUO96] Masahiro Mambo, Keisuke Usuda, and Eiji Okamoto. Proxy signatures: delegation of the power to sign messages. *IEICE Trans. Fund. of Electronic Communications and Comp Sci.* E79-A/9 (1996) 1338–1354.

[NIS91] NIST. A proposed federal information processing standard for digital signature standard (DSS). *Draft Tech. Rep. FIPS PUB XXX*, August 1991. Standards Publication (FIPS

[OO90] Tatsuaki Okamoto, Kazuo Ohta. Divertible zero-knowledge interactive proofs and commutative random self-reducibility. *Eurocrypt '89* LNCS 434, Springer-Verlag 1990, 134–149.

[RSA78] Ronald L. Rivest, Adi Shamir, Leonhard Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *CACM* 21/2 (1978) 120-126, reprinted: 26/1 (1983) 96–99.

[Sim83] Gustavus J. Simmons. A "Weak" Privacy Protocol Using the RSA Crypto Algorithm. *Cryptologia* 7/2 (1983) 180–182.

[Sim84] Gustavus J. Simmons. The Prisoners' Problem and the Subliminal Channel. *Crypto '83.* Plenum Press, New York 1984, 51–67.