Associative One-Way Functions: A New Paradigm for Secret-Key Agreement and Digital Signatures

Muhammad Rabi and Alan T. Sherman*
Computer Science Department
University of Maryland Baltimore County
Baltimore, Maryland 21228-5398
email: mrabi@cs.umbc.edu and sherman@cs.umbc.edu

November 15, 1993

Abstract

We propose *associative one-way functions* as a new cryptographic paradigm for exchanging secret keys and for signing digital documents. First, we precisely define these functions and establish some of their basic properties. Next, generalizing a theorem of Selman, we constructively prove that they exist if and only if $P \neq NP$. In addition, we exhibit an implementation based on integer multiplication. We present a novel protocol that enables two parties to agree on a secret key, and we discuss the security of this protocol. Finally, we generalize our protocol to enable two or more parties to agree on a secret key, and we present a similar protocol for signing documents.

Given any honest binary function $\circ : \mathcal{S} \times \mathcal{S} \rightarrow \mathcal{S}$ on the message space $\mathcal{S} = \{0,1\}^*$, we say that $\circ$ is *associative one-way* if: 1) for all $x, y, z \in \mathcal{S}$, $x \circ (y \circ z) = (x \circ y) \circ z$; and 2) $\circ$ is polynomial-time computable, but inverting $\circ$ is not. We say that $\circ$ is *strong* if inverting $x \circ y$ remains hard when given x or y.

With any strong associative one-way function $\circ$ on the message space $\mathcal{M} = \{0,1\}^n$, Alice and Bob can agree on a secret key in $\mathcal{M}$ as follows. 1) Alice selects $x, y \in \mathcal{M}$ at random and sends $x \circ y$ and y to Bob. 2) Bob selects $z \in \mathcal{M}$ at random and sends $y \circ z$ to Alice. 3) Alice computes $k_A = x \circ (y \circ z)$ and Bob computes $k_B = (x \circ y) \circ z$. The associativity of $\circ$ implies $k_A = k_B$, which value Alice and Bob adopt as their secret key.

For digital signatures, Alice selects a pair of numbers $x_A, y_A \in \mathcal{M}$ at random, places $(y_A, x_A \circ y_A)$ in an authenticated public directory, and keeps x_A secret. To sign any message $m \in \mathcal{M}$, Alice computes $m \circ x_A$. To verify the message-signature pair $(m, m \circ x_A)$, Bob obtains y_A and $x_A \circ y_A$ from the public directory and checks if $(m \circ x_A) \circ y_A = m \circ (x_A \circ y_A)$.

Keywords. Associative one-way functions, complexity theory, cryptology, cryptography, crypto-complexity, cryptographic protocols, digital signatures, key-agreement problem, secret-key exchange, public-key cryptography.

*Part of this work was carried out while Sherman was a member of the Institute for Advanced Computer Studies, University of Maryland College Park.

1 Introduction

Secret-key agreement is a fundamental problem in communications security that is of significant interest to practitioners and theorists. In its simplest form, this problem is for two parties (say, Alice and Bob), to agree on a secret key k by sending messages back and forth over an insecure channel which is wiretapped by a passive eavesdropper (say, Eve). The main security requirement is that Eve must not have more than an ϵ-advantage in guessing k listening to the communications over guessing k without listening to the communications, where ϵ is some small positive real number (say, $\epsilon = 0.01$). We present a new solution to this problem based on the novel paradigm of associative one-way functions.

Our paradigm combines two crucial properties from algebra and cryptography—associativity (of function application) and one-wayness (of a cryptographic function). Let $\mathcal{M}$ be any message space; for example, a practitioner might choose $\mathcal{M} = \{0,1\}^n$ for some positive integer n. An *associative one-way function (AOWF)* on $\mathcal{M}$ is any binary function $\circ : \mathcal{M} \times \mathcal{M} \to \mathcal{M}$ that is both associative and one-way. Function $\circ$ is associative if, for all $x, y, z \in \mathcal{M}$, $x \circ (y \circ z) = (x \circ y) \circ z$. Intuitively, $\circ$ is one-way if $\circ$ is easy to compute but hard to invert. We say that $\circ$ is *strong* if inverting $x \circ y$ is hard even when given x or y.

In this paper we present two results involving AOWFs: First, extending Selman's [16] complexity-theoretic framework for studying one-way functions, we formalize our notion of an AOWF and constructively prove that one exists if and only if $P \neq NP$. We also give a practical implementation of an AOWF for which inverting it is as hard as factoring. Second, we present an elegant two-round protocol that applies any strong AWOF to solve the two-party secret-key agreement problem. Using an information-theoretic framework suggested by Yao [19, 20], we discuss the security of the protocol. In addition, we extend the protocol to handle two or more parties, and we present a similar protocol for signing documents. Throughout, we focus on unauthenticated key agreement; additional mechanisms are needed if Bob is to convince himself that he has been communicating with Alice.

Given the state-of-the-art in cryptology, we distinguish between plausible and provable implementations and solutions. A *provable* implementation of an AOWF is an implementation that, under suitable intractability assumptions (*e.g.* factoring is hard), is shown to satisfy the definition of an AOWF. By contrast, a *plausible* implementation appears through intuitive arguments to satisfy the definitions. For example, we consider the RSA cryptosystem a plausible implementation of a public-key cryptosystem since no one has proven that breaking it is equivalent to factoring.

The key agreement problem was introduced in 1976 by Diffie and Hellman [5, 6], who referred to it as the public-key distribution problem. Without any proof of security, Diffie and Hellman also suggested a plausible solution based on the difficulty of computing discrete logarithms over $GF(p)$. Since then, several other approaches to key agreement have been proposed. For example, in 1988 McCurley [13] put forth a method and proved it secure, provided integer factoring is hard. Motivated by practical engineering considerations for "Clipper Chip" hardware, in 1993 Leighton and Micali [12] devised a new way to achieve key agreement based on hash functions and physically-secure memories. Additional methods have been suggested in 1987 by Blom [1], in 1992 by Diffie, van Oorschot, and Wiener [7], and in 1992 by Blundo, De Santis, Herzberg, Kutten, Vaccaro, and Yung [2]. Although several approaches to proving the security of key-agreement protocols have been tried (*e.g.* see Desmedt and Burmester [4] and van Oorschot [14]), no one has proven the security of the Diffie-Hellman scheme, even assuming that computing discrete logarithms is infeasible in polynomial time.

Any public-key cryptosystem can be used to effect secret-key agreement, but practitioners have sought solutions that do not require or make available the additional mechanisms and capabilities of public-key cryptosystems. Our approach to key agreement offers a new cryptographic building block—AOWFs—that can also be applied to sign documents.

The idea of associative one-way functions is due to Sherman, who proposed the concept in 1984 in his exploration of relationships among algebraic and security properties of cryptographic functions [17, 11]. Its application in the key-agreement protocol was suggested by Rivest and Sherman in 1984. We expect that the intriguing AOWF concept will be shown to have many other useful applications in cryptology.

The rest of this paper is organized in six sections. Section 2 precisely defines the concept of an AOWF and proves two basic properties about them. Section 3 proves that AWOFs exist if and only if $P \neq NP$. Section 4 gives a provable implementation of an AOWF. Section 5 presents two applications of AOWFs: a new key-agreement protocol and a new digital-signature protocol. Section 6 discusses the security of the key-agreement protocol. Finally, Section 7 summarizes our conclusions and lists several open problems.

2 AOWFs: Definitions and Basic Properties

An *associative one-way function (AOWF)* is any binary function that is associative and one-way. In this section, we precisely define these functions (in their strong and weak forms) and prove two basic properties about them. We formulate our definitions by extending the crypto-complexity framework of Selman [16]. In this framework, a one-way function is any function that can be computed in polynomial time but that cannot be inverted in polynomial time.

2.1 Definitions

Within our definitions, we shall deal exclusively with binary functions on the infinite message space $\mathcal{S} = \{0,1\}^*$ of all finite binary strings. Let $\circ : \mathcal{S} \times \mathcal{S} \rightarrow \mathcal{S}$ be any such function. We will denote function application using infix or prefix notation (*e.g.* $x \circ y = \circ(x,y)$). Throughout, unless otherwise stated, all functions are total (*e.g.* domain$(\circ) = \mathcal{S} \times \mathcal{S}$). Although the definitions are straightforward, there are some important technicalities (*e.g.* honesty requirement) and subtleties (*e.g.* dealing with inverses and partial functions). For any strings $x, y \in \mathcal{S}$, let $|x|$ denote the length (*i.e.* number of characters) of x, and let $x||y$ denote the concatenation of x and y.

To ensure that the difficulty of inverting an AOWF not be caused simply by its input being much longer than its output, we require every AOWF to be honest in the following standard sense.

Definition 1 Any binary function $\circ : \mathcal{S} \times \mathcal{S} \rightarrow \mathcal{S}$ is *honest* if and only if there exists a polynomial p such that, for every $z \in$ image$(\circ)$, there exist $x, y \in \mathcal{S}$ such that $x \circ y = z$ and $|x| + |y| \leq p(|z|)$.

Because we do not require that AOWFs be injective (in fact, we prove they cannot be), we must explain what it means to invert a non-injective function. By inverting $\circ$ we mean: given any $z \in$ image$(\circ)$, find *any* $x, y \in \mathcal{S}$ such that $x \circ y = z$.

Definition 2 Any binary function $\circ : \mathcal{S} \times \mathcal{S} \rightarrow \mathcal{S}$ is *one-way* if and only if $\circ$ is honest; $\circ$ is computable in polynomial time; and inverting $\circ$ is not computable in polynomial time.

Our key agreement protocol requires its AOWF to be difficult to invert, even if either one of its inputs is given. By inverting $\circ$ given its second argument, we mean inverting the restricted function $\circ_y = \circ(\cdot, y)$; that is, given any

$y \in \mathcal{S}$ and any $z \in \text{image}(\circ_y)$, find any $x \in \mathcal{S}$ such that $x \circ y = z$. Inverting $\circ$ given its first argument is similarly defined. The reader may find it helpful to draw an analogy with the security requirements of a conventional encryption function, which can be viewed as a binary function that combines a key and a plaintext to produce a ciphertext. In this analogy, a strong one-way function corresponds to an encryption function that is secure against a known-plaintext attack.

Definition 3 Any binary one-way function $\circ : \mathcal{S} \times \mathcal{S} \to \mathcal{S}$ is *strong* if and only if inverting $\circ$ given its first argument is not computable in polynomial time and inverting $\circ$ given its second argument is not computable in polynomial time.

By associativity, we shall always mean associativity of function application. Since our existence proof for total AOWFs first establishes the existence of partial AOWFs, we need to extend the usual notion of associativity to partial functions.

Definition 4 Let $\circ : \mathcal{S} \times \mathcal{S} \to \mathcal{S}$ be any partial binary function. We say $\circ$ is *associative* if and only if $x \circ (y \circ z) = (x \circ y) \circ z$. If $\circ$ is total, we require this equation to hold for all $x, y, z \in \mathcal{S}$. If $\circ$ is partial, we require this equation to hold for all $x, y, z \in \mathcal{S}$ such that each of (x, y), (y, z), $(x, y \circ z)$, and $(x \circ y, z)$ is an element of domain$(\circ)$.

Combining Definitions 1–4 yields our definition of an AOWF.

Definition 5 Any binary function $\circ : \mathcal{S} \times \mathcal{S} \to \mathcal{S}$ is a *(strong) AOWF* if and only if $\circ$ is both associative and (strongly) one-way.

2.2 Basic Properties

We now prove two basic facts about AOWFs. First, we prove that no AOWF is injective. Second, we give a sufficient condition for which any multiplicative one-way function can be easily converted into an AOWF.

Proposition 1 *No AOWF is injective.*

Proof (By contradiction). Suppose there existed some injective AOWF $\circ : \mathcal{S} \times \mathcal{S} \to \mathcal{S}$. Given any $z \in \mathcal{S}$, we could compute a preimage of z in constant time as follows. By the associativity of $\circ$, for any $y \in \mathcal{S}$, $z \circ (y \circ z) = (z \circ y) \circ z$. By the injectivity of $\circ$, it would follow that $z = z \circ y$ and $y \circ z = z$. Thus, (z, y) and (y, z) would be preimages of z, contradicting the one-wayness of $\circ$. $\square$

To construct an AOWF, one plausible way would be to convert an existing (unary) one-way function into an AOWF. Proposition 2 gives a sufficient two-part condition for when this conversion can be carried out. The first part of this condition states that the one-way function $f : G \to G$ be an endomorphism of some abelian semigroup $\mathcal{G} = (G, *)$; thus, f is multiplicative in the sense that $f(a * b) = f(a) * f(b)$ whenever $a, b \in G$. The second part requires $a * f(b) = b * f(a)$ whenever $a, b \in G$. The constructed AOWF is simply the functional composition of f and $*$. If G contains an identity element, then it is easy to prove that this construction produces a strong AOWF.

Proposition 2 *Let $\mathcal{G} = (G, *)$ be any abelian semigroup; let $f : G \to G$ be any multiplicative one-way function on G; and define $\circ : G \times G \to G$ by $a \circ b = f(a * b)$ whenever $a, b \in G$. If, for all $a, b \in G$, $a * f(b) = b * f(a)$, then $\circ$ is an AOWF.*

Proof (Direct proof). We must prove that $\circ$ is one-way and associative. *One-wayness:* Let $z \in \text{image}(\circ)$. Any preimage (a, b) of z under $\circ$ yields the preimage $a * b$ of z under f. Therefore, since f is one-way, so is $\circ$. *Associativity:* Let $a, b, c \in G$. We must show that $a \circ (b \circ c) = (a \circ b) \circ c$. By the definition of $\circ$, and by the multiplicativity of f, it is true that $a \circ (b \circ c) = f(a * f(b * c)) = f(a) * f(f(b)) * f(f(c))$ and $(a \circ b) \circ c = f(f(a * b) * c) = f(f(a)) * f(f(b)) * f(c)$. Furthermore, the hypothesis implies $f(a) * f(f(c)) =$

$f(f(a)) * f(c)$. The desired result follows from the commutativity and associativity of $*$. $\square$

3 Existence Proof

Generalizing a theorem of Selman [16], we constructively prove that AOWFs exist if and only if $P \neq NP$. Under the hypothesis $P \neq NP$, our main theorem proves the existence of a partial AOWF. Our construction is based on the computation tree of any polynomial-time nondeterministic Turing machine that accepts any language in $NP - P$. We begin by reviewing Selman's theorem.

Brassard was one of the first researchers to investigate connections between the existence of one-way functions and relationships among complexity classes. In 1979, Brassard [3] proved that the existence of a one-way function would imply $P \subsetneq \Delta$ and thus $P \neq NP$, where $\Delta = NP \cap CoNP$. Brassard required that his one-way function be bijective and that its image be in $CoNP$. In 1984, Grollmann and Selman [10][16, Proposition 3] tightened this result by proving that an *injective* one-way function exists if and only if $P \neq UP$, where UP (unique P) is the class of languages that can be accepted in nondeterministic polynomial time with unique accepting paths.[1] Dropping the injectivity requirement, Selman [16] also proved that one-way functions exist if and only if $P \neq NP$. Thus, the existence of an injective one-way function depends on a stronger (and apparently much stronger) complexity assumption than does the existence of a one-way function. Since AOWFs cannot be injective (see Proposition 1), we shall focus on Selman's second theorem.

Theorem 1 (Selman 1992). *There exists a one-way function if and only if $P \neq NP$.*

Proof (By construction). See Selman [16], Proposition 1. To prove the sufficiency condition, Selman considers any language $A \in NP - P$ and any NP-machine M that accepts A. He constructs a one-way function as the inverse of the function $comp_M : \mathcal{S} \to \mathcal{S}$, defined for any $x \in \mathcal{S} = \{0,1\}^*$ as follows. If $x \in A$, then $comp_M(x)$ is any accepting configuration of M on input x; otherwise, $comp_M(x) = \perp$. Intuitively, $comp_M^{-1}$ is one-way because it is is easy to traverse M's computation tree upwards but hard to traverse this tree downward. In particular, it is hard to decide if $x \in A$. $\square$

To extend Theorem 1 to AOWFs, we modify the *comp* function so that its inverse is a binary associative function. Our idea for this modification comes from a graphical interpretation of the definition of associativity.

Theorem 2 *There exists a partial AOWF if and only if $P \neq NP$.*

Proof (Necessity and sufficiency).
($\Rightarrow$) Since every AOWF is a one-way function, the proof follows from Theorem 1.
($\Leftarrow$) Assume $P \neq NP$. Then there exists some language $A \in NP - P$. Let M be any NP-machine that accepts A, and let $\mathcal{C}_M$ denote the set of all configurations of all computations of M. We will construct a partial AOWF as any inverse of the function $acomp_M : \mathcal{C}_M \to \mathcal{C}_M \times \mathcal{C}_M$, which we will now define.

First, for any $x \in \mathcal{C}_M$, define the predicate $\Phi_M(x)$ to be true if and only if there exist some string $w \in A$ and some configurations $y_0, y_1 \in \mathcal{C}_M$ in the computation tree of M on input w such that $x \notin \{y_0, y_1\}$ and x is the closest common ancestor of y_0 and y_1. Then, for any $x \in \mathcal{C}_M$, define

$$acomp_M(x) = \begin{cases} (y_0, y_1) & \text{if } \Phi_M(x) \text{ is true} \\ \perp & \text{otherwise,} \end{cases} \tag{1}$$

where (y_0, y_1) is any pair of configurations as described in the definition of $\Phi_M(x)$. It is possible that $y_0 = y_1$.

Now, define the partial function $f : \mathcal{C}_M \times \mathcal{C}_M \to \mathcal{C}_M$ to be any inverse of $acomp_M$.

[1] According to Selman [16], the essence of this result was independently discovered by Berman in 1977.

We will prove that f is honest; f is associative, f is computable in polynomial time; and f cannot be inverted in polynomial time.

1. Claim: f is honest. We must show that there exists some polynomial p such that, for all $x \in \mathcal{C}_M$, $|acomp_M(x)| \leq p(|x|)$. This inequality holds for p being twice the running time of M. It is true that M runs in polynomial time, and no configuration can be larger than the time needed to compute it. Thus, f is honest.

2. Claim: f is associative. Let x, y, z be any configurations in $\mathcal{C}_M$ such that each of (x, y), (y, z), $(x, f(y, z))$, and $(f(x, y), z)$ is an element of domain(f). By the definition of associativity, we must prove $f(x, f(y, z)) = f(f(x, y), z)$; that is, we must prove $f(w_0, z) = f(x, w_1)$, where $w_0 = f(x, y)$ and $w_1 = f(y, z)$. By the definition of f, there exists some $w \in A$ such that w_0 is the closest common ancestor of x and y, and w_1 is the closest common ancestor of y and z, along some computation paths in the computation tree of M on input w. It follows that $f(w_0, z) = f(x, w_1)$ since this configuration is the closest common ancestor of w_0 and w_1.

3. Claim: f is computable in polynomial time. Let (y_0, y_1) be any configurations in domain(f). Thus, there exists $w \in A$ such that y_0 and y_1 are configurations along some computation paths in the computation tree of M on input w. Since M runs in polynomial time, these paths are at most polynomially long. By traversing these paths upwards, $f(y_0, y_1)$ can be computed in polynomial time as the closest common ancestor of y_0 and y_1. Hence, f is computable in polynomial time, even though recognizing domain(f) might take longer.

4. Claim: f^{-1} is not computable in polynomial time. To compute f^{-1} is to compute $acomp_M$. Were $acomp_M$ polynomial-time computable, we could decide A in polynomial time as follows. Given any input string b, let x_b be any child of the initial configuration of the computation of M on input b. Then $b \in A$ if and only if $acomp_M(x_b) \neq \perp$. Since $A \notin P$, $acomp_M$ is not computable in polynomial time. □

We observe that the AOWF f constructed in the proof of Theorem 2 is commutative, because the closest common ancestor relation is commutative. This function, however, is not a strong AOWF because, given any $x, y_1 \in \mathcal{C}_M$ such that x is in the image of f restricted to second argument y_1, it follows that $f(y_1, y_1) = x$. Although recognizing domain(f) is as hard as recognizing A, given any AOWF g with domain$(g) \in P$, it is possible to extend g to a total AOWF $\hat{g}$ in a straightforward fashion. Specifically, let $c \in \mathcal{C}_M$ be any string such that $(c, c) \notin$ domain(g). Then define $\hat{g}(x, y) = g(x, y)$ whenever $(x, y) \in$ domain(g), and $\hat{g}(x, y) = c$ otherwise.

4 Implementations

Although Theorem 2 constructively proves the existence of AOWFs (assuming $P \neq NP$), our construction does not lead to a natural implementation. Therefore, in this section, we present three concrete examples of practical AOWFs based on integer multiplication, matrix multiplication, and logical OR. In Section 5.1, we explain why the Diffie-Hellman [5] scheme based on modular exponentiation does not yield an AOWF. The problem of finding a strong AOWF remains open.

Integer and Matrix Multiplication

Integer multiplication over the large odd integers is a provable AOWF. This operation is associative and easy to compute. Moreover, its inverse problem is integer splitting, which is believed to be hard. This operation is also commutative. Integer multiplication, however, is not a strong AOWF: for any positive integers $x, y > 1$, given xy and y, it is easy to compute x by integer division. Similarly, multiplication of square integer matrices is a provable AOWF.[2]

[2] Although these functions are not strong AOWFs in the adult cryptographic world, they are examples of strong AOWFs in Fellow's [9] world of kid krypto. Division of large integers is intractable for most first graders,

Logical OR

An alternate type of strong AOWF is the bitwise logical OR function $\vee : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ defined on binary strings of some fixed length n. This function is not one-way in our sense because any output bit with value 0 must be produced by $0 \vee 0$ and any output bit with value 1 can be produced by $1 \vee 1$. Nevertheless, this function is associative and commutative and offers some information-theoretic protection for some of its inputs, as we shall now explain.

Let $a, b \in \{0,1\}$ and let $c = a \vee b$. When $c = 1$, it is not possible from c alone to distinguish between the three possible cases $a, b = 1, 1$ and $a, b = 0, 1$ and $a, b = 1, 0$. Moreover, given c and b, if $b = c = 1$, it is not possible to determine the value of a. Thus, if x, y are chosen at random from $\{0,1\}^n$, given $x \vee y$ and y, we expect $n/2$ bits of x to remain unknown. Specifically, these hidden bits are those in the positions where the y string contains a one.

5 Applications

Through their combination of associativity and one-wayness, AOWFs provide an elegant mechanism for secret-key agreement and digital signatures. In this section we present novel protocols that use any strong AOWF for two-party secret-key agreement and any strong commutative AOWF for multi-party secret key agreement. In addition, we explain how any strong AOWF can be applied to sign messages. We also present an implementation of our two-party key agreement protocol using discrete logarithms, and we generalize our protocol and the Diffie-Hellman protocol to enable multi-party key agreement. In Section 6 we discuss the security of these protocols.

and Gaussian elimination and matrix inversion is typically infeasible through sixth grade.

5.1 Key Agreement

Protocol KAP given below shows how Alice and Bob can agree on a secret key from the set $\mathcal{M} = \{0,1\}^n$, where n is some positive integer.

1. Alice generates two random numbers x and y. Alice keeps x secret and sends y and $x \circ y$ to Bob.
2. Bob generates a random number z. Bob keeps z secret and sends $y \circ z$ back to Alice.
3. Alice computes $k_A = x \circ (y \circ z)$ and Bob computes $k_B = (x \circ y) \circ z$. Alice and Bob agree on $k_A = k_B$ as their secret key.

The protocol exploits a strong AOWF $\circ : \mathcal{M} \times \mathcal{M} \to \mathcal{M}$, known to Alice, Bob, and the enemy Eve. During the protocol, random numbers x, y, z are selected from the set $\mathcal{M}$. In the last step of the protocol, Alice computes her key $k_A = x \circ (y \circ z)$ and Bob computes his key $k_B = (x \circ y) \circ z$; the associativity of $\circ$ ensures that $k_A = k_B$. Figure 1 summarizes this procedure.

An Implementation of Protocol KAP Using Discrete Logarithms

Any implementation of a strong AOWF gives an implementation of Protocol KAP. Although we have shown examples of provable AOWFs, we have not found any example of a plausible strong AOWF. Therefore, in this section, we give an alternate implementation of protocol KAP without using an AOWF. This implementation modifies the Diffie-Hellman [5] key agreement scheme to produce a function $\diamond$, which although not an AOWF, can be used to implement Protocol KAP. Along the way, we explain why the Diffie-Hellman scheme does not yield an AOWF.

The basis of the Diffie-Hellman key-exchange scheme can be viewed as computing the binary function $\psi : \mathbb{Z}_p \times \mathbb{Z}_p \to \mathbb{Z}_p$ defined by $\psi(g, x) = g^x \bmod p$ whenever $x \in \mathbb{Z}_p$, where p is some large prime integer and g is a primitive element modulo p. This function is a one-way function since it is easy to compute and since there is no known polynomial-time algorithm for computing discrete logarithms modulo p. Moreover, this function is strong in its first argument because, in the discrete logarithm problem, g is known. But the function is not associative because it is not true that $g^{(x^y)} = (g^x)^y \bmod p$ for all $x, y, z \in \mathbb{Z}_p$.

In the Diffie-Hellman scheme, after selecting p and g, Alice picks $x \in \mathbb{Z}_p$ at random and sends $\psi(g,x)$, g, and p to Bob. Next, Bob picks $y \in \mathbb{Z}_p$ at random and sends $\psi(g,y)$ to Alice. Finally, Alice computes $\psi(\psi(g,y),x) = (g^y)^x \bmod p$ and Bob computes $\psi(\psi(g,x),y) = (g^x)^y \bmod p$, which value they adopt as their secret key. This scheme depends not on associativity but on the fact that $\psi(\psi(g,y),x) = \psi(\psi(g,x),y)$ for all $x, y \in \mathbb{Z}_p$.

To implement Protocol KAP using the discrete logarithm, let p be any large prime integer; let $D = \mathbb{Z}_p \times \mathbb{Z}_2$; and define the binary function $\diamond : D \times D \to D$ as follows. Given any $x, y \in D \times D$, let $x = (a_x, b_x)$ and $y = (a_y, b_y)$ and define

$$x \diamond y = \begin{cases} (a_y^{a_x} \bmod p, 0) & \text{if } b_x = 1 \text{ and } b_y = 1 \\ (a_y^{a_x} \bmod p, 1) & \text{if } b_x = 0 \text{ and } b_y = 1 \\ (a_x^{a_y} \bmod p, 1) & \text{if } b_x = 1 \text{ and } b_y = 0 \\ (a_x^{a_y} \bmod p, 0) & \text{if } b_x = 0 \text{ and } b_y = 0. \end{cases} \tag{2}$$

To use $\diamond$ in Protocol KAP, Alice selects $x = (a_x, 1)$ and $y = (g, 1)$, where g is a primitive element modulo p. Bob selects $z = (a_z, 0)$. It follows that $x \diamond y = (g^{a_x} \bmod p, 0)$; $y \diamond z = (g^{a_z} \bmod p, 1)$; $x \diamond (y \diamond z) = ((g^{a_x})^{a_z} \bmod p, 0)$; and $(x \diamond y) \diamond z = ((g^{a_z})^{a_x} \bmod p, 0)$. Thus, for these values of x, y, z it is true that $x \diamond (y \diamond z) = (x \diamond y) \diamond z$, even though $\diamond$ is not associative.

Multi-Party Key Agreement

Whenever the strong AOWF $\circ$ is commutative, Protocol KAP can be generalized to enable any N parties to agree on a secret key. In this generalized protocol, which we shall call *GKAP*, Party 1 selects $y \in \mathcal{M}$ at random and sends y to all other parties. Next, each party i selects $x_i \in \mathcal{M}$ at random, keeps x_i secret, and sends $x_i \circ y$ to all other parties. Finally, each party i computes the key $k_i = x_i \circ (x_1 \circ y) \circ (x_2 \circ y) \circ \cdots \circ (x_{i-1} \circ y) \circ (x_{i+1} \circ y) \circ \cdots \circ (x_N \circ y)$. By the associativity and commutativity of $\circ$, each of these keys equals $x_1 \circ x_2 \circ \cdots \circ x_N \circ y^{N-1}$, where $y^{N-1} = (y \circ \cdots \circ y)$ denotes $N-2$ applications of $\circ$.

We note that the Diffie-Hellman scheme can be similarly generalized. Thus, given any large prime integer p and any primitive element g modulo p, each party i selects $x_i \in \mathbb{Z}_p$ at random. The protocol works in N stages, and it is convenient to think of the N parties as arranged in a ring. At stage $1 \leq j < N$, participant i receives a value y_j from party $i-1 \bmod N$. Initially, $y_1 = g$. Party i computes $y_j^{x_i} \bmod p$ and sends this value to party $i+1 \bmod N$. At the Nth stage, each party i computes $k_i = y_N^{x_i} \bmod p$ and agrees on this value as the secret key. Thus, the agreed upon key is $g^{x_1 x_2 \cdots x_N} \bmod p$. To the best of our knowledge, this generalization of the Diffie-Hellman protocol has not been previously suggested.

5.2 Digital Signatures

As illustrated in Figure 2, strong AOWFs can also be used to sign messages. To this end, let $\circ : \mathcal{M} \times \mathcal{M} \to \mathcal{M}$ be any strong AOWF, and assume that there is an authenticated public directory. Initially, each user U generates two numbers $x_U, y_U \in \mathcal{M}$ at random, keeps x_U secret, and places the pair $(y_U, x_U \circ y_U)$ into the public directory. To sign any message $m \in \mathcal{M}$, the user computes the signature $\sigma_U(m) = m \circ x_U$. To verify any purported message-signature pair (m, σ) from U, the recipient retrieves y_U and $x_U \circ y_U$ from the public directory and computes $\sigma \circ y_U$ and $m \circ (x_U \circ y_U)$. The recipient accepts σ as a valid signature of m by U if and only if $\sigma \circ y_U = m \circ (x_U \circ y_U)$.

As with many other signature schemes, this scheme is vulnerable to what Rivest [15] calls existential forgery: given any valid message-signature pair $(m, \sigma_U(m))$, it is possible to forge signatures of new messages of the form $m' = z \circ m$, for any $z \in \mathcal{M}$. Specifically, forge $\sigma_U(m') = m' \circ x_U = (z \circ m) \circ x_U$ by computing $\sigma_U(m') = z \circ \sigma_U(m) = z \circ (m \circ x_U)$. To overcome this difficulty, one could use a public cryptographically-secure hash function, as is typically done in the Elgamal signature scheme [8]. When using a hash function $h : \mathcal{M} \rightarrow \mathcal{M}$, the signer would compute the signature $h(m) \circ x_U$ and assume that Eve cannot find any $z \in \mathcal{M}$ and any intelligible message $m' \in \mathcal{M}$ such that $h(m') = z \circ h(m)$.

6 Security of Protocol KAP

In this section we discuss the security of Protocol KAP by stating what is means for this protocol to be secure and by observing some of its properties. In addition, we discuss how one can reason about the security of KAP in Yao's [19, 20] model of computational information theory. Although we have been unable to prove or disprove the security of KAP, no one has proven or disproven the security of the Diffie-Hellman scheme either.

Discussion

At the end of the protocol Eve knows y, $x \circ y$, and $y \circ z$. For the protocol to be secure, from this information, Eve must not be able to guess the agreed-upon key $x \circ y \circ z$ with advantage. Our assumptions are that $\circ$ is a strong AOWF on $\mathcal{M} = \{0,1\}^n$ and that x, y, z are chosen independently with uniform distribution from $\mathcal{M}$.

If Eve could compute x or z, then Eve could compute the key as $x \circ (y \circ z)$ or $(x \circ y) \circ z$. This direct attack is impossible because it would contradict the assumption that $\circ$ is a strong AOWF. Thus, the only way in which KAP could possibly fail is if Eve could compute $x \circ y \circ z$ without computing x or z. Equivalently, if the only way to compute $x \circ y \circ z$ from y, $x \circ y$, and $y \circ z$ were to compute x or z, then KAP would be secure.

One possible attack would be to compute $x \circ y \circ z$ by applying $\circ$ to some sequence of terms drawn from the given values y, $x \circ y$, and $y \circ z$. For example, if $y^r = y$ for some $r - 1$ applications of $\circ$, then Eve could compute $x \circ y \circ z = (x \circ y) \circ y^{r-2} \circ (y \circ z)$. Thus, for KAP to be secure, it must not be true that $y^r = y$ for some polynomially-bounded r. But this condition is excluded by the fact that $\circ$ is a strong AOWF.

Security of KAP in Yao's Model

One elegant model in which to reason about the security of KAP is Yao's model of computational information theory, which distinguishes between accessible and inaccessible information with respect to polynomial-time computations. In this section we briefly and intuitively discuss how one might do so.

Let X, Y, Z be independent random variables with uniform distribution over $\mathcal{M}$. Let $S = (X, Z)$ denote the secret information; let $K = X \circ Y \circ Z$ denote the agreed-upon key; and let $G = (Y, X \circ Y, Y \circ Z)$ denote the information given to Eve. More formally we would need to introduce additional technical details (*e.g.* a polynomial distribution ensemble); for clarity and brevity we omit these details, which can be easily added.

In Yao's model, we say that $\circ$ is strongly one-way (in its second argument) if the conditional effective entropy $H_C(X|Y, X \circ Y)$ of X given Y and $X \circ Y$ is large in the following sense.

Definition 6 The operator $\circ$ is *computationally information-theoretically strongly one-way* if and only if $H_C(X|Y, X \circ Y) \in \Omega(1/r^t)$ and $H_C(Z|Y, Y \circ Z) \in \Omega(1/r^t)$, for some fixed t.

Intuitively, this conditional effective entropy is the amount of uncertainty (in bits) that the polynomially-bounded Eve can compute about X given Y and $X \circ Y$. Here, r denotes the number of observed samples of the random variables.

Similarly, we say that Protocol KAP is secure if the effective entropy $H_C(K)$ and conditional effective entropy $H_C(K|G)$ are approximately equal in the following sense.

Definition 7 Protocol KAP is *computationally information-theoretically secure* if and only if $H_C(K) - H_C(K|G) \in O(v(r))$.

In Definition 7, for any function g, the notation $g(n) \in O(v(r))$ means $g(r) \in O(1/r^t)$ for every positive integer t.

From our new definition of strongly one-way, we prove that Eve cannot deduce useful information about X or Z in Protocol KAP.

Proposition 3 *If $\circ$ is computationally information-theoretically strongly one-way, then $H_C(S|G) \in \Omega(1/r^t)$.*

Proof (Direct proof). By the definition of conditional effective mutual information, $H_C(S|G) = H_C(X|G) + H_C(Z|G) - I_C(S|G)$, where $I_C(S|G)$ is the conditional effective mutual information of S given G. By the strong one-wayness of $\circ$, there exists a positive integer t such that $H_C(X|G) \in \Omega(1/r^t)$ and $H_C(Z|G) \in \Omega(1/r^t)$. Moreover, $I_C(S|G) = 0$ because X and Z are independent. Therefore, $H_C(S|G) \in \Omega(1/r^t)$. □

Whether it is possible to extend this proof to the security of KAP remains open.

7 Conclusion and Open Problems

We have introduced the concept of associative one-way functions (AOWFs) as an intriguing and useful new cryptographic paradigm. We proved that AOWFs exist if and only $P \neq NP$, and we presented protocols for applying strong AOWFs to reach unauthenticated secret-key agreement and to sign documents.

AWOFs illustrate a beneficial synergism that can ensue when a cryptographic object is endowed with a combination of algebraic and security properties. To explore such combinations, AOWFs are a natural place to start because they combine two of the most fundamental properties from algebra and cryptographic security: associativity and one-wayness.

We conclude with eight open problems. 1) Do strong AOWFs exist? What complexity-theoretic assumptions (*e.g.* $P \neq NP$) imply the existence of strong AOWFs? What relationships exist among the existence of strong AOWFs and the existence of other cryptographic objects? 2) Exhibit a plausible strong AOWF. 3) Under reasonable intractability assumptions (*e.g.* factoring integers or computing discrete logarithms is hard), exhibit a provable strong AOWF. 4) Given any one-way function, construct a strong AOWF from it. 5) Prove (or disprove) that protocol KAP is secure. 6) What can be said about the distribution of the agreed-upon key in protocol KAP?[3] In particular, is it possible to modify Protocol KAP to ensure that neither Alice nor Bob alone can bias the distribution of the agreed-upon key? 7) What other applications do AOWFs have? 8) What other combinations of algebraic and security properties of cryptographic functions yield useful cryptographic objects?

Although the security of Protocol KAP remains open, so does that of the Diffie-Hellman protocol. Nevertheless, Protocol KAP and our digital signature method are evidence that AOWFs—and more generally, functions that combine fundamental algebraic and security properties—offer elegant solutions to a variety of practical cryptographic problems.

[3] For some related work on the Diffie-Hellman scheme, see Waldvogel and Massey [18].

Acknowledgments

We thank Richard Chang for helpful editorial comments.

References

[1] Blom, R., "An optimal class of symmetric key generation systems" in *Advances in Cryptology: Proceedings of Crypto 84*, Springer-Verlag (1987), 335–338.

[2] Blundo, C., A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly secure key distribution for dynamic conferences" in *Advances in Cryptology: Proceedings of Crypto 92*, Springer-Verlag, to appear.

[3] Brassard, Gilles, "A note on the complexity of cryptography," *IEEE Transactions on Information Theory*, **IT-25**:2 (March 1979), 232–233.

[4] Desmedt, Yvo, and Mike Burmester, "Towards practical 'proven secure' authenticated key distribution" in *Proceedings of the 1st ACM Conference on Computer and Communications Security*, ACM Press (1993), 228–231.

[5] Diffie, Whitefield, and Martin E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, **IT-22**:6 (November 1976), 644–654.

[6] Diffie, Whitfield, and Martin E. Hellman, "Privacy and authentication: An introduction to cryptography," *Proceedings of the IEEE*, **IT-67**:3 (March 1979), 397–427.

[7] Diffie, Whitfield, Paul C. van Oorschot, and Michael J. Wiener, "Authentication and authenticated key exchanges," *Designs, Codes and Cryptography*, **2**:2 (June 1992), 107–125.

[8] Elgamal, Taher, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, **IT-31**:4 (July 1985), 469–472.

[9] Fellows, Michael R., and Neal Koblitz, "Kid krypto" in *Proceedings of Crypto 92*, Springer-Verlag, to appear.

[10] Grollman, J., and A. Selman, "Complexity measures for public-key cryptosystems," *SIAM Journal on Computing*, **17**:2 (1988), 309–335.

[11] Kaliski, Burton S. Jr., Ronald L. Rivest, and Alan T. Sherman, "Is the Data Encryption Standard a group? (Results of cycling experiments on DES)," *Journal of Cryptology*, **1**:1 (1988), 3–36.

[12] Leighton, Tom, and Silvio Micali, "New approaches to secret-key exchange (extended abstract)" in *Advances in Cryptology: Proceedings of Crypto 93*, Springer-Verlag, to appear.

[13] McCurley, Kevin S., "A key distribution system equivalent to factoring," *Journal of Cryptology*, **1**:2 (1988), 95–105.

[14] van Oorschot, Paul C., "Extending cryptographic logics of belief to key agreement protocols (extended abstract)" in *Proceedings of the 1st ACM Conference on Computer and Communications Security*, ACM Press (1993), 232–243.

[15] Rivest, Ronald L., "Cryptography" in *Handbook of Theoretical Computer Science. Volume A: Algorithms and Complexity*, J. van Leeuwen, ed., Chapter 13, MIT Press/Elsevier (1990), 717–755.

[16] Selman, Alan L., "A survey of one-way functions in complexity theory," *Mathematical Systems Theory*, **25**:3 (1992), 203–221.

[17] Sherman, Alan T., "Cryptology and VLSI (a two-part dissertation)," Technical Report MIT/LCS/TR-381, MIT Laboratory for Computer Science (October 1986).

[18] Waldvogel, C. P., and J. L. Massey, "The probability distribution of the Diffie-Hellman key" in *Proceedings of AUSCRYPT 92*, to appear.

[19] Yao, Andrew C., "Theory and applications of trapdoor functions (extended abstract)" in *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science* (November 1982), 80–91.

[20] Yao, A. C., "Computational information theory" in *Complexity in Information Theory*, Yaser S. Abu-Mostafa, ed., Chapter I, Springer-Verlag (1988), 1–15.

Appendix A: Figures

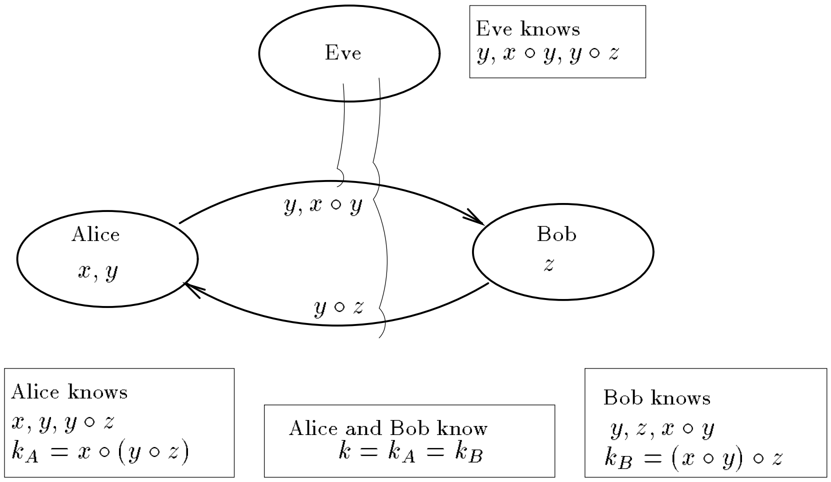

Figure 1: Pictorial view of key-agreement Protocol KAP, which applies a strong associative one-way function $\circ : \mathcal{M} \times \mathcal{M} \to \mathcal{M}$. Alice sends y and $x \circ y$ to Bob (keeping x secret), and Bob sends $y \circ z$ to Alice (keeping z secret). At the end of the protocol, the passive eavesdropper Eve knows the values y, $x \circ y$, and $y \circ z$, since each of these values was sent over the insecure transmission line. The one-wayness of the associative function $\circ$ ensures that, from these values, the eavesdropper cannot deduce x or z.

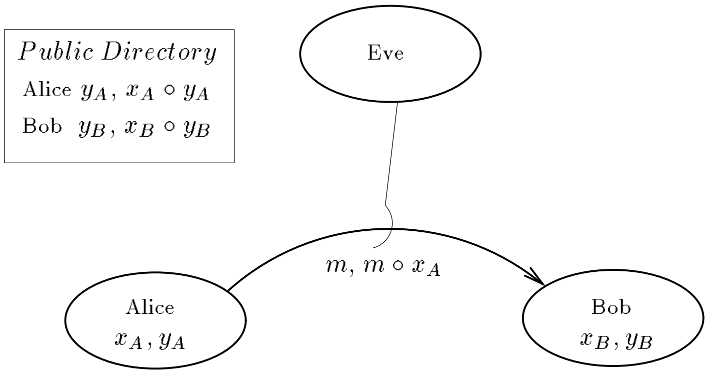

Figure 2: Pictorial view of a procedure for signing messages using a strong associative one-way function $\circ : \mathcal{M} \times \mathcal{M} \rightarrow \mathcal{M}$. Initially, Alice places her public information $y_A, x_A \circ y_A$ in an authenticated public directory. To sign any message m, Alice computes the signature $m \circ x_A$ using her secret information x_A. To verify the message-signature pair $(m, m \circ x_A)$, Bob checks if $m \circ (x_A \circ y_A) = (m \circ x_A) \circ y_A$ using the public information $y_A, x_A \circ y_A$.