Computer Security
Lecture 3

Cryptography I:
Crypto Foundations

David Aspinall

School of Informatics,

University of Edinburgh.

16th January 2003

http://www.informatics.ed.ac.uk/teaching/modules/cs

2003/04/17 16:08:04.

Contents

Cryptography may be part of a security solution, but it is never the whole solution. At best, it transforms a more general security problem into a key management problem.

Definitions and Goals

Cryptography has a long history. Its original application, and still the most common one, is to enable two parties to communicate in secret, across a unsecured (public) channel.

- **cryptography** — the science of secret writing, using *ciphers*
- **cryptanalysis** — the science of breaking ciphers
- **cryptology** — both of above
- **encryption** — transforming *plain text* into *cipher text*
- **decryption** — recovering *plain text* from *cipher text*
- **e'n scheme**, **cipher**, **cryptosystem** — a mechanism for encryption

Cryptography can be used to help ensure these security properties:

- **confidentiality** — preventing open access to information
- **integrity** — preventing unauthorized modification of data
- **authentication** — verification of identity
- **non-repudiation** — preventing denial of previous actions

Cryptographic primitives

We want to ensure the security properties mentioned, even when another party may eavesdrop or intercept messages. The use of carefully designed **cryptographic protocols** helps this. Protocols are built using a small number of **cryptographic primitives**, parameterised on 0, 1, or 2 **keys**.

Unkeyed Primitives	Secret-key Primitives	Public-key Primitives
Hash functions	Symmetric-key ciphers	Public-key ciphers
MD5, RIPEMD-160, SHA-1	— block and stream	RSA, ElGammal
One-way permutations	3DES, Blowfish, Rijndael (AES)	Digital signatures
Random sequences	Keyed hash functions (MACs)	DSA
	Identification primitives	Identification primitives
	Digital signatures	
	Pseudorandom sequences	

Security and cryptography: two views

We've pointed out already that there can be no single absolute notion of what "security" means. When studying systems using cryptography, we can take one of two views:

- **Perfect crypto primitives**
  - Crypto primitives are operators in an abstract data type.
  - Operators assumed perfect (e.g., encryption impossible to break).
  - Other assumptions (e.g., key text differentiable from cipher text)
  - Used for formal models of security protocol correctness. Correctness statements are relative to assumptions about primitives.
- **Realistic security notions for crypto primitives**
  - Consider particular implementations (MD5, DES, etc.)
  - Analyse design of cryptosystem (security, "strength") and particular algorithms (security, efficiency)
  - Cryptosystems have different notions of security (information-theoretic, complexity-theoretic, probabilistic, . . .)

Two issues in crypto primitive security

A proper analysis of security primitives must include considering *modes of attack*. However, there are two common issues which are worth mentioning up-front.

Openness vs security-by-obscurity

- Since Kerckhoffs' desiderata (1883), it has been understood that for keyed ciphers, *security should lie wholly in the key*. ("Compromise of the system details should not incovenience the correspondents")
- Nowadays, it's usually recommended that cryptosystems have an *open design* so that they can reviewed by as many experts as possible. Many examples of where security-by-obscurity has failed.

Key size in encryption systems

- It's necessary *but not sufficient* to have a key space large enough to prevent a feasible *brute force* attack (exhaustive search).
- Rule-of-thumb: a key space of 2^{80} is currently considered large enough. But this is a *very simplistic* view; we'll go further later.

Functional foundations: bijections

- Recall that a **bijection** is a mathematical function which is one-to-one (injective) and onto (surjective).
- In particular, if $f : X \to Y$ is a bijection, then for all $y \in Y$, there is a unique $x \in X$ such that $f(x) = y$. This unique x is given by the *inverse* function $f^{-1} : Y \to X$.

Bijections are used as the basis of cryptography, for encryption. If f is an encryption transformation, then f^{-1} is the corresponding decryption transformation.

Why restrict to bijections? If a non-injective function was used as as an encryption transformation, it would not be possible to decrypt to a unique plain text.

(Saying this, non-bijections, in fact non-functions, are used as encryption transformations. Can you imagine how?)

Message spaces

We assume:

- A set $\mathcal{M}$, the *message space*.
  $\mathcal{M}$ consists of symbol strings, e.g., binary strings, English text, etc.
  Elements $m \in \mathcal{M}$ are called *plaintexts*.
- A set $\mathcal{C}$, the ciphertext space.
  $\mathcal{C}$ also consists of strings of symbols.
  Elements $c \in \mathcal{C}$ are called *ciphertexts*.
- Each space is given over some *alphabet*, a set $\mathcal{A}$.
  For example, we may consider $\mathcal{A}$ to be the letters of the English alphabet A-Z, or the set of binary digits $\{0, 1\}$. (Of course, any alphabet can be encoded using words over $\{0, 1\}$).

Cryptography systems

- An *encryption transformation* is a bijection $E : \mathcal{M} \to \mathcal{C}$.
- A *decryption transformation* is a bijection $D : \mathcal{C} \to \mathcal{M}$.

Encryption and decryption transformations are often indexed using *keys*.

- The *key space* is a finite set $\mathcal{K}$. Elements $k \in \mathcal{K}$ are called *keys*.
- An **cryptography system** (aka encryption scheme or cipher) consists of two sets indexed by keys
  - a family of encryption functions $\{E_e \mid e \in \mathcal{K}\}$
  - a family of decryption functions $\{D_d \mid d \in \mathcal{K}\}$

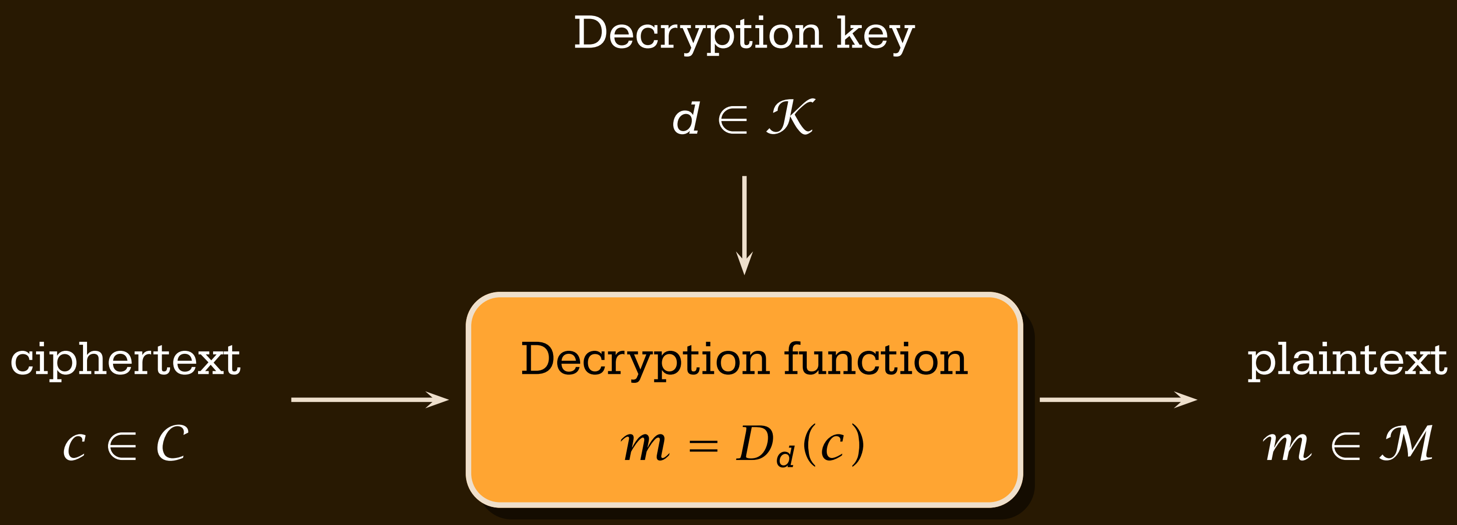
  such that for each $e \in \mathcal{K}$, there is a unique $d \in \mathcal{K}$ with $D_d = E_e^{-1}$. We call such a pair (e, d) a *key pair*.

Encryption

$$\begin{tikzcd}
 & \begin{array}{c}\text{Encryption key}\\ e \in \mathcal{K}\end{array} \arrow[d] & \\
\begin{array}{c}\text{plaintext}\\ m \in \mathcal{M}\end{array} \arrow[r] & \boxed{\begin{array}{c}\text{Encryption function}\\ c = E_e(m)\end{array}} \arrow[r] & \begin{array}{c}\text{ciphertext}\\ c \in \mathcal{C}\end{array}
\end{tikzcd}$$

When discussing protocols later, we'll use another notation for encryption: $\{M\}_{K_{ab}}$ stands for the message M encrypted under the key K_{ab}. We write K_{ab} to indicate a key shared by parties A and B. When we use this notation, the corresponding decryption operation is not mentioned.

Decryption



Decryption key
$d \in \mathcal{K}$
ciphertext
$c \in \mathcal{C}$
Decryption function
$m = D_d(c)$
plaintext
$m \in \mathcal{M}$

Symmetric and asymmetric cryptography

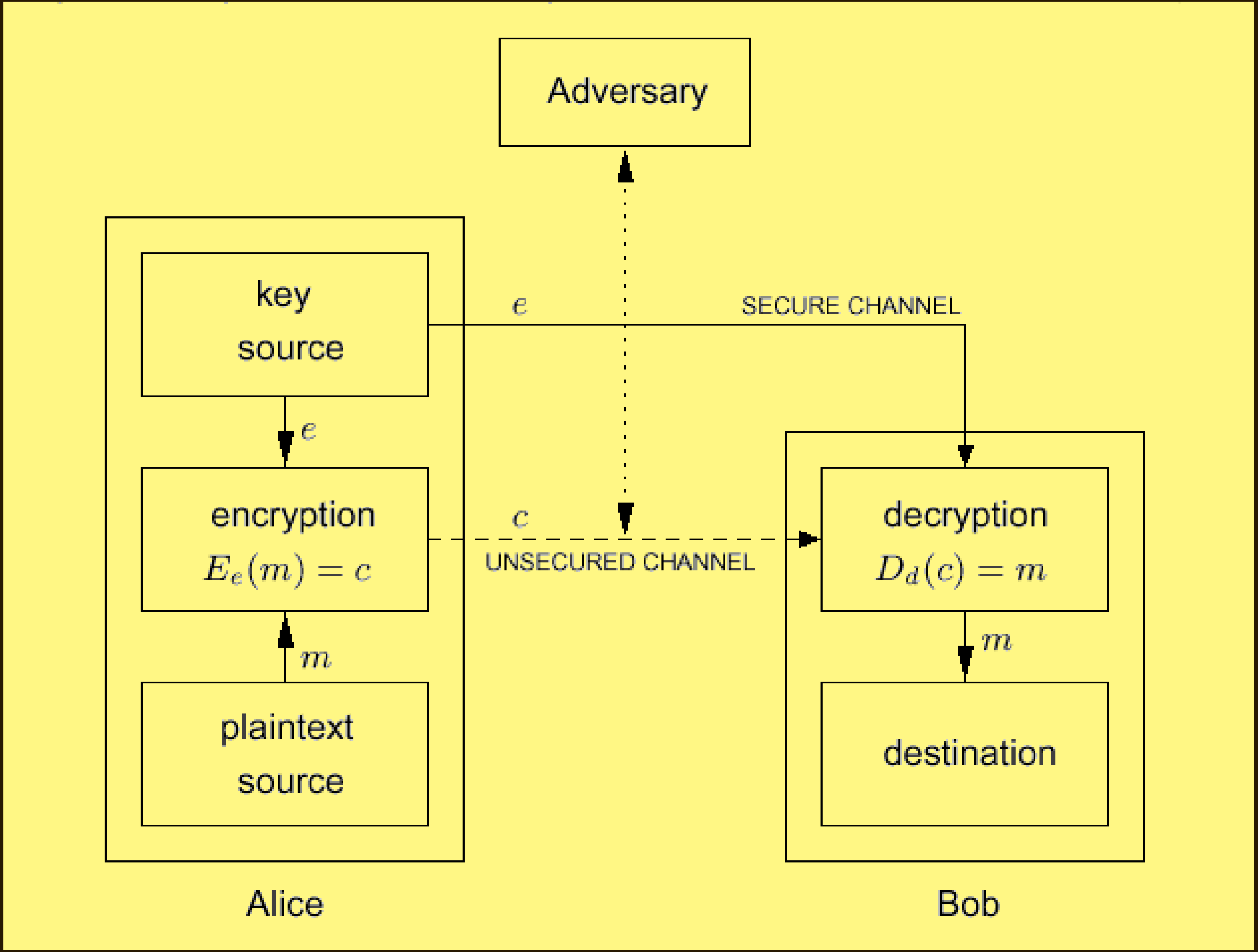
Now we can define symmetric and asymmetric cryptography:

- **symmetric** cryptography
  - *e* and *d* are (essentially) the same
  - aka secret-key, shared-key, single-key, conventional
- **asymmetric** cryptography
  - Given *e*, it is (computationally) infeasible to find *d*.
  - aka public-key (PK), since *e* can be made public.

These definitions are a bit imprecise: to be exact, one should define the meanings of "essentially" in the first case and "computationally infeasible" in the second case.

NB: the key-pair relation is not the only difference we notice; other differences are characteristics of the algorithms we know for each form of cryptography, and because of the way each form is used.

Symmetric cryptography



Asymmetry: a ground breaking discovery!

- Although we've started with a framework which builds in the ideas of public key cryptography, we shouldn't forget how truly ground breaking its discovery was.
- Secure channels are difficult and expensive to implement. The problem of delivering secret keys through unsecured channels had confounded cryptographers, governments, and other authorities for many centuries. But:

  If you can read everything I write, I cannot rely on any secret that has gone before, how can I possibly send a confidential message to my friend which you cannot also understand?
- The answer uses a creative leap of innovation (two keys, one public), as well relying on some clever maths in its implementation (*trapdoor one-way functions*).

Foundations: one-way functions

- A function $f : X \rightarrow Y$ is called a **one-way function** if
  - — it is feasible to compute $f(x)$ for all $x \in X$, but
  - — it is infeasible to find any x in the pre-image of f, such that $f(x) = y$, for a randomly chosen $y \in \mathrm{Im}\, f$.
    (If f is bijective, this means it is infeasible to compute $f^{-}1(y)$).

By definition, a one-way function is not useful for encryption. But it may be useful as a *cryptographic* or *one-way* hash function.

However, the definition above is vague: to be exact, we should give precise notions of *feasible* and *infeasible*. This is possible, but so far **no-one has proved the existence of a true one-way function**. Some functions used in modern ciphers are properly called *candidate one-way functions*, which means that there is a body of belief that they are one-way.

Trapdoor one-way functions

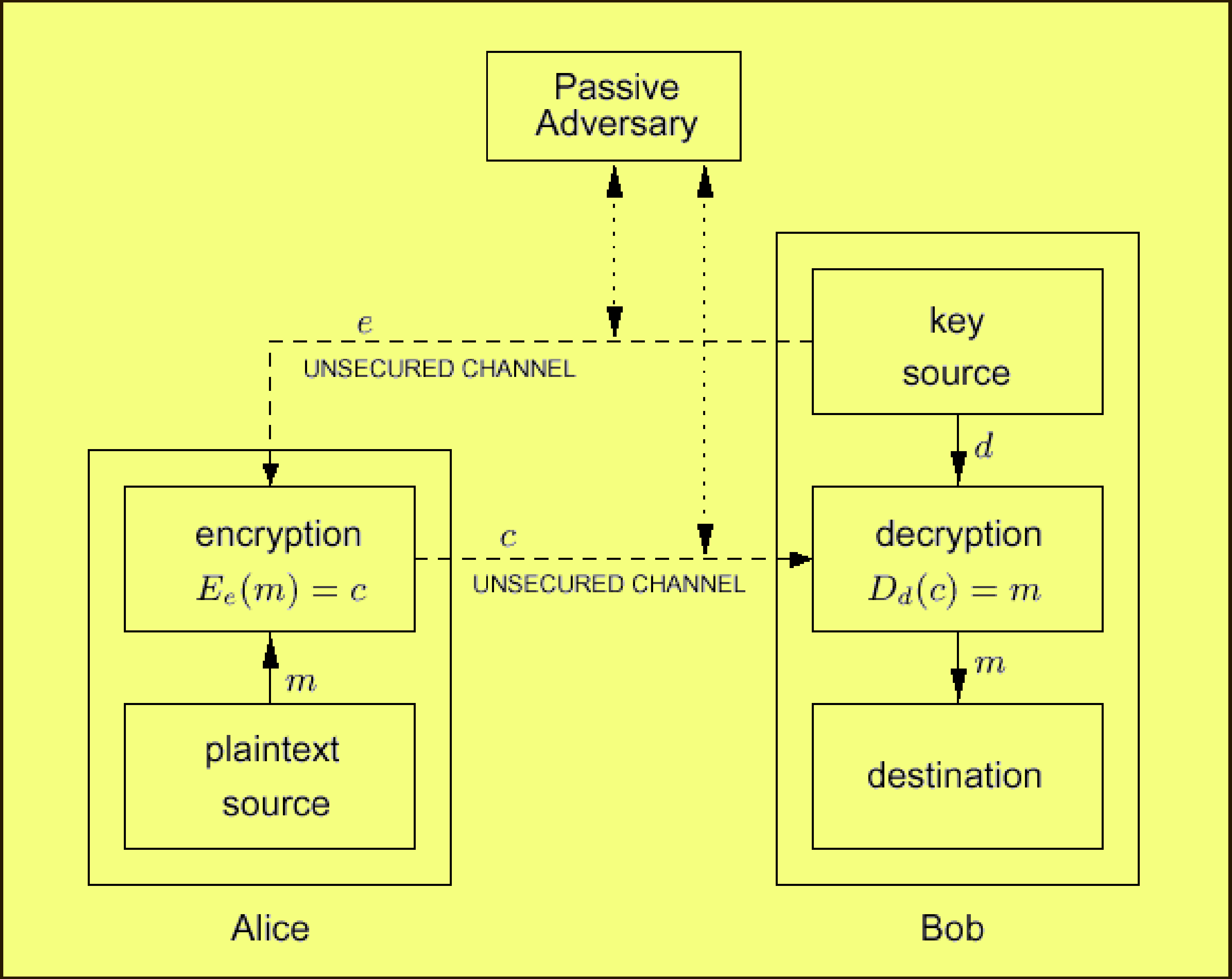
- A **trapdoor one-way function** is a one-way function f that has a "trapdoor": given some additional information, it becomes feasible to compute an x such that $f(x) = y$, for random $y \in \operatorname{Im} f$.

Trapdoor one-way functions are just what we need for public key cryptography: the trapdoor information is provided by the private key.

Again, we know some likely candidates, but no functions have yet been formally proved to be trapdoor one-way functions. So there is always **a possibility of breaking present cryptography systems by discovering new algorithms** based on advances in mathematics and cryptanalysis. It's considered unlikely that one-way functions do *not* exist (some hash functions are as provably secure as NP-complete problems).

For cipher functions in common use, catastrophic failure is also considered unlikely. Instead, we see gradual failure due to advances in computation power and (non-revolutionary but clever) improvements in algorithms or cryptanalysis, bringing some attacks closer to feasibility.

Asymmetric cryptography



Properties of hash functions

- A *hash function* is a **computationally efficient** function $h : \{0,1\}^* \to \{0,1\}^k$ which **compresses** binary strings of arbitrary length into k-length binary *hash-values*.
- A good hash function distributes hash values uniformly: the probability that a randomly chosen string s gets mapped to a particular hash value y is $\frac{1}{2^k}$
- A *cryptographic hash function* has some further properties, e.g.,
  1. **pre-image resistance (one-way)**: given a hash-value y, it is computationally infeasible to find an x such that $h(x) = y$
  2. **2nd pre-image resitance**: given a value x_1 and its hash $h(x_1)$, it is computationally infeasible to find an x_2 such that $h(x_2) = h(x_1)$.
  3. **collision resistance**: it is computationally infeasible to find any two inputs x_1 and x_2 such that $h(x_1) = h(x_2)$.

A *one-way hash function* satisfies 1,2; a *collision resistant hash function* satisfies 2, 3 (& usually 1). Note that 3$\Rightarrow$2.

Cryptographic hash functions

- Hash functions are also known as *modification detection codes* (the MD in MD5).
- Their primary use is to provide *data integrity*, especially where *malicious* alteration is a concern, e.g. software distribution. (Ordinary hash functions such as CRC-checkers produce *checksums* which are not pre-image resistant: an attacker could produce a hacked version of a software product and ensure the checksum remained the same).
- *Message Authentication Codes (MACs)*. MACs are keyed hash functions; just like hash functions but indexed with a secret key. As well as data integrity, they provide data-origin authentication (because it is assumed that apart from the recipient, only the sender knows the secret key).

Digital signatures

Digital signatures provide a means for a principal to bind (a representation of) its identity to a piece of information. A digital signature mechanism for a principal A is given by:

- A message space $\mathcal{M}$ of messages for signing
- A set $\mathcal{S}$ of *signatures* (e.g. fixed length binary strings)
- A secret *signing function* $S_A : \mathcal{M} \to \mathcal{S}$
- A public *verification function* $V_A : \mathcal{M} \times \mathcal{S} \to \mathsf{Bool}$

such that these correctness and security properties hold:

1. $V_A(m, s) = \mathsf{true}$ if and only if $S_A(m) = s$.
2. For any principal other than A, it is computationally infeasible to find for any $m \in \mathcal{M}$, an $s \in \mathcal{S}$ such that $V_A(m, s) = \mathsf{true}$.

We usually use a public algorithm yielding key-indexed families $\{S_s \mid s \in \mathcal{K}\}$ of signing and $\{V_v \mid v \in \mathcal{K}\}$ of verification functions.

Once more, nobody has formally proved a signature system exists...

Digital signatures from PK encryption

Suppose we have a public-key encryption scheme with $\mathcal{M} = \mathcal{C}$, and (d, e) a key-pair. Then because E_e and D_d are both permutations on $\mathcal{M}$, we have that:

$$D_d(E_e(m)) \;=\; E_e(D_d(m)) \;=\; m \quad \text{for all } m \in \mathcal{M}$$

A public-key scheme of this type is called *reversible*. (RSA is reversible, but it's important to remember that not every PK scheme is).

We can define a digital signature scheme thus, by reversing the roles of encryption and decryption (recall that d is private, e is public):

- $\mathcal{M}$ is the message space, $\mathcal{C} = \mathcal{M}$ the signature space;
- the signing function $S_A = D_d$
- the verification function V_A is defined by

$$V_A(m, s) = \begin{cases} \text{true} & \text{if } E_e(s) = m, \\ \text{false} & \text{otherwise.} \end{cases}$$

However, this scheme is somewhat too simple. . .

Digital signatures using redundancy

- The previous scheme is too simple because signatures are forgeable: a principal B can generate a random $s \in \mathcal{S}$ as a signature, apply the public encryption function to get a message $m = E_e(s)$, and transmit (m, s). This will verify, and is an example of *existential forgery*, in that the message m is not likely to be of B's choosing (and probably garbage). But this ability violates property 2 given on slide 20.
- Common fix: take $\mathcal{M}' \subset \mathcal{M}$ to be messages with a special (redundant) structure, which is publicly known e.g., messages padded to an even length, and surrounded with a fixed bit pattern. This format can be easily recognized by the verifier, which now does:

$$V_A(s) = \begin{cases} \text{true} & \text{if } E_e(s) \in \mathcal{M}', \\ \text{false} & \text{otherwise.} \end{cases}$$

  Now A only needs to transmit the signature s, since the message $m = E_e(s)$ can be recovered by the verification function. This property is *message recovery*. Existential forgery is less likely.

Key management

- **Key management** concerns the establishment and maintenance of keying relationships between principals.
- Key establishment concerns *agreement* and *transport* mechanism. Maintenance concerns the *renewal* of keys.
- For symmetric crypto, each pair of principals requires a shared key. This leads to needing $\binom{n}{2} = \frac{n(n-1)}{2}$ or $O(n^2)$ keys. This gets to be a very large number of keys on a network with many hosts.
- Symmetric crypto solution: use a **trusted third party** (TTP) which shares a key with each principal (n keys), and will communicate shared keys as necessary.
- Public key crypto: use a central directory of public keys. Problem: active adversaries may masquerade by replacing public keys. Solution: use a TTP to sign keys, binding keys and ownership details together to generate *digital certificates*.

References

Some of the content in this lecture is adapted from Chapter 1 of [MOV97]. Bruce Schneier's popular text [Sch96] gives a readable treatment of the subject, without too much maths. Nigel Smart's book [Sma03] is more rigorous and up-to-date. A detailed history of cryptography is given in [Kah97].

[Kah97] David Kahn. *The Codebreakers*. Simon & Schuster, revised edition, 1997.

[MOV97] Alfred J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone, editors. *Handbook of Applied Cryptography*. CRC Press Series on Discrete Mathematics and Its Applications. CRC Press, 1997. Online version at http://cacr.math.uwaterloo.ca/hac.

[Sch96] Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, second edition, 1996.

[Sma03] Nigel Smart. *Cryptography: An Introduction*. McGraw-Hill, 2003.

Crypto usability is still one of the biggest challenges the industry has to solve and our new relationship with the FIO is all about trying to find a solution to this big issue. CEO of Bitcoin.com. Stefan Rust.Â We are very excited to be a part of the Foundation for Interwallet Operability. This is a valuable opportunity to address and solve cross-wallet usability issues that plague the industry. CEO of Infinito. Why Invest in Crypto Funds? The crypto market is still at its very earl.Â This is a list of crypto funds which invest and trade in cryptocurrencies and blockchain companies.The fees list are in percent and per year. In some cases, the fees have been simplified, and the maximum fee has been listed. Why Invest in Crypto Funds? The crypto market is still at its very early stages. No one knows which cryptocurrency or utility token will be at the top of itâ€™s niche in five years time. The CryptoRights Foundation, Inc. (CRF) is a 501(c)(3) non-profit organization based in San Francisco. The CryptoRights Foundation helps human rights groups and other NGOs use encryption to protect their online communications. It has contributed to encryption standards such as OpenPGP, IPsec and GnuPG. The organization was founded on 26 February 1998 during a total solar eclipse (near maximum totality off the active volcanic Caribbean island of Montserrat) on a boat chartered by attendees of the Â© 2018 by Crypto Foundation Proudly created with Wix.com. FACEBOOK (Coming soon). TWITTER (Coming soon). INSTAGRAM (Coming soon). ABOUT US >. Under the Japanese Act on General Incorporated Associations and General Incorporated Foundations. Crypto Foundation. Home. About. More.