Cryptography I: Introduction Computer Security Lecture 3

Mike Just¹

School of Informatics University of Edinburgh

18th January 2009

¹Based on original lecture notes by David Aspinall

Outline

Terminology

Basic Definitions

Symmetric Cryptography

Asymmetric Cryptography

Outline

Terminology

Basic Definitions

Symmetric Cryptography

Asymmetric Cryptography

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

cryptography: science of secret writing with ciphers

- cryptography: science of secret writing with ciphers
- cryptanalysis: science of breaking ciphers

- cryptography: science of secret writing with ciphers
- cryptanalysis: science of breaking ciphers
- cryptology: both of above

- cryptography: science of secret writing with ciphers
- cryptanalysis: science of breaking ciphers
- cryptology: both of above
- encryption: transforming plain text to cipher text

- cryptography: science of secret writing with ciphers
- cryptanalysis: science of breaking ciphers
- cryptology: both of above
- **encryption**: transforming *plain text* to *cipher text*
- decryption: recovering plain text from cipher text

- cryptography: science of secret writing with ciphers
- cryptanalysis: science of breaking ciphers
- cryptology: both of above
- encryption: transforming plain text to cipher text
- decryption: recovering plain text from cipher text
- encryption scheme, cipher, cryptosystem: a mechanism for encryption and decryption

Cryptography can be directly used to help ensure these security properties:

Cryptography can be directly used to help ensure these security properties:

confidentiality — preventing open access

Cryptography can be directly used to help ensure these security properties:

- confidentiality preventing open access
- integrity preventing unauthorized modification

Cryptography can be directly used to help ensure these security properties:

- confidentiality preventing open access
- integrity preventing unauthorized modification
- authentication verification of identity Sometimes split into:
 - entity authentication
 - data origin authentication

Cryptography can be directly used to help ensure these security properties:

- confidentiality preventing open access
- integrity preventing unauthorized modification
- authentication verification of identity Sometimes split into:
 - entity authentication
 - data origin authentication
- non-repudiation preventing denial of actions

We want to ensure these properties, even when another party may eavesdrop or intercept messages.

Carefully designed cryptographic protocols help this.

Cryptographic primitives

Protocols are built using **cryptographic primitives**, parametrised on 0, 1, or 2 **keys**.

Unkeyed	Secret key	Public key
Random sequences One-way permutations Hash functions	Symmetric-key ciphers — block and stream Keyed hash functions (aka MACs) Identification primitives Digital signatures Pseudorandom sequences	Public-key ciphers Digital signatures Identification primitives

Familiar examples:

- Hash functions: MD5, SHA-1, SHA-256
- Symmetric block ciphers: DES, 3DES, AES
- Public key ciphers: RSA, El Gammal
- Digital signature schemes: RSA, DSA

- Hash functions h(m)
 - integrity: "fingerprint" provides tamper evidence
 - message compression: hash-then-sign schemes

- Hash functions h(m)
 - integrity: "fingerprint" provides tamper evidence
 - message compression: hash-then-sign schemes
- Symmetric block ciphers $E_k(m)$, $D_k^{-1}(m)$
 - bulk encryption: network comms, data storage

- Hash functions h(m)
 - integrity: "fingerprint" provides tamper evidence
 - message compression: hash-then-sign schemes
- Symmetric block ciphers $E_k(m)$, $D_k^{-1}(m)$
 - bulk encryption: network comms, data storage
- Public key (asymmetric) ciphers $E_e(m)$, $D_d(m)$
 - key exchange: establishing shared keys for symmetric ciphers

- Hash functions h(m)
 - integrity: "fingerprint" provides tamper evidence
 - message compression: hash-then-sign schemes
- Symmetric block ciphers $E_k(m)$, $D_k^{-1}(m)$
 - bulk encryption: network comms, data storage
- Public key (asymmetric) ciphers $E_e(m)$, $D_d(m)$
 - key exchange: establishing shared keys for symmetric ciphers
- ► Digital signature schemes $S_A(m)$, $V_A(m, s)$
 - key signing: public key infrastructures (PKIs)

Choosing primitives

Choice of primitives influenced by:

- functionality needed
- performance
- implementation ease
- degree of security
- Degree of security is tricky: may consider
 - primitives are "perfect", maybe "unbreakable"
 - what is the worst that can happen?
 - primitives are "imperfect"
 - what does attacker know?
 - how much effort can attacker spend?

Assume perfect cryptography primitives

Primitives are operators in an abstract data type.

- Primitives are operators in an abstract data type.
- Operators are perfect (cannot break encryption).

- Primitives are operators in an abstract data type.
- Operators are perfect (cannot break encryption).
- Other assumptions, e.g., key text differentiable from cipher text.

- Primitives are operators in an abstract data type.
- Operators are perfect (cannot break encryption).
- Other assumptions, e.g., key text differentiable from cipher text.
- Used for formal analysis of security protocol correctness. Correctness statements are relative to assumptions about primitives.

Assume perfect cryptography primitives

- Primitives are operators in an abstract data type.
- Operators are perfect (cannot break encryption).
- Other assumptions, e.g., key text differentiable from cipher text.
- Used for formal analysis of security protocol correctness. Correctness statements are relative to assumptions about primitives.

Assume perfect cryptography primitives

- Primitives are operators in an abstract data type.
- Operators are perfect (cannot break encryption).
- Other assumptions, e.g., key text differentiable from cipher text.
- Used for formal analysis of security protocol correctness. Correctness statements are relative to assumptions about primitives.

Model real cryptography primitives

Attacker knowledge may allow cryptanalysis

Assume perfect cryptography primitives

- Primitives are operators in an abstract data type.
- Operators are perfect (cannot break encryption).
- Other assumptions, e.g., key text differentiable from cipher text.
- Used for formal analysis of security protocol correctness. Correctness statements are relative to assumptions about primitives.

- Attacker knowledge may allow cryptanalysis
- Consider specific algorithms (MD5, DES, etc.).

Assume perfect cryptography primitives

- Primitives are operators in an abstract data type.
- Operators are perfect (cannot break encryption).
- Other assumptions, e.g., key text differentiable from cipher text.
- Used for formal analysis of security protocol correctness. Correctness statements are relative to assumptions about primitives.

- Attacker knowledge may allow cryptanalysis
- Consider specific algorithms (MD5, DES, etc.).
- Analyse design of cryptosystem (security, "strength") and algorithms (security, efficiency).

Assume perfect cryptography primitives

- Primitives are operators in an abstract data type.
- Operators are perfect (cannot break encryption).
- Other assumptions, e.g., key text differentiable from cipher text.
- Used for formal analysis of security protocol correctness. Correctness statements are relative to assumptions about primitives.

- Attacker knowledge may allow cryptanalysis
- Consider specific algorithms (MD5, DES, etc.).
- Analyse design of cryptosystem (security, "strength") and algorithms (security, efficiency).
- Study cryptographic notions of security (information-theoretic, complexity-theoretic, probabilistic, ...).

Cryptanalysis attacks

Setup: have $c_1 = E_k(m_1), \dots, c_n = E_k(m_n)$ for small n.

- Best outcome: find k or algorithm for D_k^{-1} .
- Try to better brute-force (exhaustive search).

Attack type	Attacker knowledge
Ciphertext only	the c_i (deduce at least m_i)
Known plaintext	the c_i and m_i
Chosen plaintext	c_i for chosen m_i
Adaptive chosen plaintext	as above, but iterative
Chosen ciphertext	$c_i, m_i = D_d(c_i)$. Find decryption key d .
"Rubber-hose"	bribery, torture, or blackmail
"Purchase-kev"	(not cryptanalysis, but y successful)

Openness vs security-by-obscurity

Openness vs security-by-obscurity

 Kerckhoffs' desiderata (1883) recommends that for keyed ciphers, security should lie wholly in the key.
 "Compromise of the system details should not inconvenience the correspondents"

Openness vs security-by-obscurity

- Kerckhoffs' desiderata (1883) recommends that for keyed ciphers, security should lie wholly in the key.
 "Compromise of the system details should not inconvenience the correspondents"
- Nowadays, cryptosystems usually have an open design, reviewed by as many experts as possible.
 Often security-by-obscurity fails.

Openness vs security-by-obscurity

- Kerckhoffs' desiderata (1883) recommends that for keyed ciphers, security should lie wholly in the key.
 "Compromise of the system details should not inconvenience the correspondents"
- Nowadays, cryptosystems usually have an open design, reviewed by as many experts as possible.
 Often security-by-obscurity fails.

Key size in encryption systems

Openness vs security-by-obscurity

- Kerckhoffs' desiderata (1883) recommends that for keyed ciphers, security should lie wholly in the key.
 "Compromise of the system details should not inconvenience the correspondents"
- Nowadays, cryptosystems usually have an open design, reviewed by as many experts as possible.
 Often security-by-obscurity fails.

Key size in encryption systems

 Necessary but not sufficient to have a key space large enough to prevent feasible brute force attack.

Openness vs security-by-obscurity

- Kerckhoffs' desiderata (1883) recommends that for keyed ciphers, security should lie wholly in the key.
 "Compromise of the system details should not inconvenience the correspondents"
- Nowadays, cryptosystems usually have an open design, reviewed by as many experts as possible.
 Often security-by-obscurity fails.

Key size in encryption systems

- Necessary but not sufficient to have a key space large enough to prevent feasible brute force attack.
- Rule-of-thumb: a key space of 2⁸⁰ is currently considered large enough. But this is a very simplistic view!

Outline

Terminology

Basic Definitions

Symmetric Cryptography

Asymmetric Cryptography

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.

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 were used as as an encryption transformation, it would not be possible to decrypt to a unique plain text.

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 were 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 *M*, the message space.
 M holds symbol strings, e.g., binary, English.
 Elements m ∈ *M* are called *plaintexts*.

Message spaces

We assume:

- A set *M*, the message space.
 M holds symbol strings, e.g., binary, English.
 Elements m ∈ *M* are called *plaintexts*.
- A set C, the ciphertext space.
 C also consists of strings of symbols.
 Elements c ∈ C are called *ciphertexts*.

Message spaces

We assume:

- A set *M*, the message space.
 M holds symbol strings, e.g., binary, English.
 Elements m ∈ *M* are called *plaintexts*.
- A set C, the ciphertext space.
 C also consists of strings of symbols.
 Elements c ∈ C are called *ciphertexts*.
- Each space is given over some alphabet, a set A. For example, we may consider 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 $E: \mathcal{M} \rightarrow \mathcal{C}$.
- A decryption transformation is a bijection $D: C \rightarrow M$.

Cryptography systems

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

Encryption and decryption transformations are indexed using *keys*.

• The key space \mathcal{K} is a finite set of keys $k \in \mathcal{K}$.

Cryptography systems

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

Encryption and decryption transformations are indexed using *keys*.

- The key space \mathcal{K} is a finite set of keys $k \in \mathcal{K}$.
- An encryption scheme 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.

An encryption scheme is also known as a cryptography system or a cipher.

Encryption



Decryption



Outline

Terminology

Basic Definitions

Symmetric Cryptography

Asymmetric Cryptography

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.
- Of course, the key-pair relation is not the only difference between symmetric and asymmetric cryptography. Other differences arise from characteristics of known algorithms and usage modes.
- Note: these definitions are imprecise: to be exact, one should define the meanings of "essentially" and "computationally infeasible".

Symmetric cryptography



Outline

Terminology

Basic Definitions

Symmetric Cryptography

Asymmetric Cryptography

Asymmetry: a ground breaking discovery!

 Our framework builds in the ideas of public key cryptography, but we shouldn't forget how truly ground breaking its discovery was.

Asymmetry: a ground breaking discovery!

- Our framework builds in the ideas of public key cryptography, but we shouldn't forget how truly ground breaking its discovery was.
- Secure channels are difficult and costly to implement. How to deliver secret keys through unsecured channels had confounded thinkers for many centuries.

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?

Asymmetry: a ground breaking discovery!

- Our framework builds in the ideas of public key cryptography, but we shouldn't forget how truly ground breaking its discovery was.
- Secure channels are difficult and costly to implement. How to deliver secret keys through unsecured channels had confounded thinkers for many centuries.

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*).

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 ∈ Im f.
 (If f is bijective, this means it is infeasible to compute f⁻¹(y)).

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 ∈ 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.

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 ∈ 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.

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.

A trapdoor one-way function is a one-way function *f* that has a "trapdoor": given some additional information, it is feasible to compute an *x* such that *f*(*x*) = *y*, for any *y* ∈ Im *f*.

These are just what we need for public key crypto: the private key is the trapdoor information.

A trapdoor one-way function is a one-way function *f* that has a "trapdoor": given some additional information, it is feasible to compute an *x* such that *f*(*x*) = *y*, for any *y* ∈ Im *f*.

These are just what we need for public key crypto: the private key is the trapdoor information.

Again, we know *candidates*, but no function has yet been proved to be a trapdoor one-way function.

A trapdoor one-way function is a one-way function *f* that has a "trapdoor": given some additional information, it is feasible to compute an *x* such that *f*(*x*) = *y*, for any *y* ∈ Im *f*.

These are just what we need for public key crypto: the private key is the trapdoor information. Again, we know *candidates*, but no function has yet been proved to be a trapdoor one-way function.

 In principle, there is a possibility of breaking crypto systems by new algorithms based on advances in mathematics and cryptanalysis.

A trapdoor one-way function is a one-way function *f* that has a "trapdoor": given some additional information, it is feasible to compute an *x* such that *f*(*x*) = *y*, for any *y* ∈ Im *f*.

These are just what we need for public key crypto: the private key is the trapdoor information. Again, we know *candidates*, but no function has yet been proved to be a trapdoor one-way function.

- In principle, there is a possibility of breaking crypto systems by new algorithms based on advances in mathematics and cryptanalysis.
- It's unlikely that one-way functions do not exist; some hash functions are as secure as NP-complete problems.

A trapdoor one-way function is a one-way function *f* that has a "trapdoor": given some additional information, it is feasible to compute an *x* such that *f*(*x*) = *y*, for any *y* ∈ Im *f*.

These are just what we need for public key crypto: the private key is the trapdoor information. Again, we know *candidates*, but no function has yet been proved to be a trapdoor one-way function.

- In principle, there is a possibility of breaking crypto systems by new algorithms based on advances in mathematics and cryptanalysis.
- It's unlikely that one-way functions do not exist; some hash functions are as secure as NP-complete problems.
- Catastrophic failure for present functions is less common than gradual failure due to advances in computation power and (non-revolutionary but clever) algorithms or cryptanalysis, bringing some attacks closer to feasibility.

Asymmetric cryptography



References

Some content is adapted from Chapter 1 of the HAC. Schneier's text is readable (but dated). Smart's book is more rigorous. Kahn's book has a detailed history.

- A. J. Menezes, P. C. Van Oorschot, S. A. Vanstone, eds. Handbook of Applied Cryptography. CRC Press. 1997. Online: http://www.cacr.math.uwaterloo.ca/hac.
- Bruce Schneier. Applied Cryptography. John Wiley & Sons, second edition, 1996.



Nigel Smart. Cryptography: An Introduction. http://www.cs.bris.ac.uk/~nigel/Crypto_Book/



David Kahn. The Codebreakers.

Simon & Schuster, revised edition, 1997.

Recommended Reading

Chapter 1 of HAC. Chapter 3, Sections 11.1–11.2 of Smart (3rd Ed).