Guiding Principles of Effective Crypto and Security Engineering,
course notes and reading complement for Applied Crypto COMPGA12
© Nicolas T. Courtois, 2006-2009
Draft, v1.2. 02 Feb 2009

> In theory, there is no difference between theory and practice.
> But, in practice…
> [de Snepscheut, computer scientist and educator].

Foreword

This document is a mix of conventional wisdom [as known to industry developers and senior experts that worked in the security field for many years], private theories and their formulations [© Nicolas Courtois], and a few blunders [in limited supply].
It is intended to provide insights into the curious science of security and cryptology and guide developers and system architects in designing complex secure systems that will last longer than 1 year ☺. As such it attempts to anticipate the future developments of cryptology and other security technologies. It also contains a lot of thoughts and analysis about markets for security and for security products.

Markets vs. Security Technology

During your professional life you may be under the pressure to give up on the application of these guiding principles for the sake of market or some other expediency. But very early, we need to develop counter-argumentation to help us to resist this (not unusual) pressure, and this document is here to help. Security professionals need to keep doing good job, and be patient, especially if one is in advance 5,10, 20 years w.r.t. to prevailing market conditions. In the long run we can reshape the markets in make them work for our benefit.
In fact, like all institutions, markets are efficient in doing a short-term well-defined and job, and very much deficient in finding where the bright future (life, technology, markets, profits etc.) will lie. Academics do fight prevailing misconceptions and biases for the sake of it. Broader number of security professionals need yet to fight for ‘markets for security technology’ to emerge. These are too frequently ‘missing markets’ in the sense that one cannot buy security, one cannot sell security, and one cannot trade security. Most products have a placebo level of security where no serious attempt was ever made to secure them. In the meantime we need to make sure that the difference between good vs. bad technology and good vs. bad security is visible at all times. And improve our security expertise every day.

Recommended reading:
All books and blog posts by Bruce Schneier and Ross Anderson.
Book by Jacques Stern “La Science du Secret” in (French) and invited talk at Eurocrypt 2003 (in English).

What is Security?

Security[Courtois]: methods and techniques that allow to protect the value(s).

What value(s) ? All without exception (like in philosophy). Including values shared by a small number of people.

What value(s) ?

- Money [economical security]. But NOT ONLY MONEY.
- Life and the quality of life (many systems save lives)
- Privacy is a value in itself.
- Good technology vs. bad one
- Free markets [not all markets are free, many markets are ‘rigged’, other are ‘missing’],
- Freedom, justice, democracy etc…

Hegel [1770-1831], German philosophy:
„The history is the process of broadening freedom”.

Freedom, new technologies ⇒ More possibilities ⇒ More security problems…
Main ‘Meta-Theorem’ [Courtois]:

When Freedom $\to \infty$, then Security $\to 0$.

Examples:

- Invention of train/underground ⇒ threat of pickpockets
- Car traffic: Major cause of death (2M/year worldwide), trains are much safer.
- Invention of the internet ⇒ pirates, hackers, espionage, fraud, information anarchy, etc.

The goal of cryptology [Courtois]: Add security to the information technologies (IT) that are by nature insecure.

Information security ethics: what we do want / not want. Difference between good and bad is the fundament of security engineering, that cannot exist without defining what is bad/undesirable and what we want to prevent by technology. For example we may want to somewhat prevent an Internet firm from selling our private data to another firm, or at least make sure that they pay a high price for it.

Pro-active vs. Reactive Security

It is better to anticipate than to suffer the consequences.

OK, but very frequently attacks are really impossible to predict.
In fact a rational strategy is always to ignore certain threats and concentrate on the others. But priorities will change and risks are almost never correctly estimated.

The principle of fallibility [Soros]. Applies to individuals, organisations, governments, etc. Yes millions of people can be wrong, yes experts can be wrong.

Weakest Link vs. Layering the Defences

Security is only as strong as the weakest link [Schneier].
We need to avoid this situation. It doesn’t have to be so.

Defence in depth, Layering the defences.

Known in the military for thousands of years.

The Curious Science of Security

Is cryptology a science? Are cryptologists always wrong?

The ambition of cryptography is to build ``unbreakable locks'', which cannot be broken even by the combined resources and expertise of the whole planet. This is quite unlike classical ``physical'' security technologies such as safes or shields which can typically always be penetrated.

Examples: ‘open source’ crypto such as triple-DES. Never broken in the practical sense.

This is the ambition.

But…, Hard Questions, Negative and Destructive Contributions

Although certain problems in cryptography can now be solved in an elegant 'provably secure' way, most practical applications of cryptography still rely extensively arguably for a long time to come, on classical components such as block and stream ciphers with uncertain security foundations.

More generally, security is difficult, cannot be seen.

- False security solutions and false security experts are abundant. Commercial security remain very pejorative. Works if you believe it. In reality markets for security are almost totally missing (see page 1).
- Some secret government agencies are not an exception either. Obscurity is an enemy of security. It is not true that security is best learned from the military experts. As much if not more can be learned from the commercial sector and business people.

Almost all progress in security is about negative results. Positive results (constructive methods, provable security) remain a minority.

The security is difficult. Cannot be seen, only sometimes the opposite can be seen: when a product/technology is not secure.
Even then the security breach is usually hard to find and point out (it may take 20 years to find an obvious attack !)

It is not true that provable security solves this problem. It just decreases a number of possible attacks in a dramatic way but the remaining attack are subject to the same incertitudes. Attacks will always get better, they will not get worse. Major breakthroughs will occur.

Karl Popper's Philosophy of Science

[early 20th century, Karl Popper is also frequently quoted by Soros, known as the Guru of financial markets].

Most security claims should be hold as ‘provisionally' true.

- Cannot be proven.
- They can only be disproved.

Statements about security should be treated as provisionally valid until they are proven invalid.

Most cryptographic primitives (e.g. ciphers), in the absence of sound constructive engineering principles developed through progressive sophistication, elimination of weaker (broken) algorithms, and the progressive strengthening against previously known attacks. This ad-hoc methodology governs most of applied cryptology.

Karl Popper also contends that, a scientific theory (or the generalization that survives it) acquires actually more value if it was heavily and extensively tested. Similarly, the more we try to break ciphers, the greater will be the value of those that will survive. And if we don't have a serious research effort in cryptanalysis, the industry will use the cheapest ciphers they can think of, that will maybe be broken in seconds while still being used, as the example of the Oyster card shows…
AES and triple DES may be the most secure ciphers around precisely because millions of euros of research money WERE spent on trying to break them. And we need to continue trying.

Is Cryptography a Science?

Science is not different from religion, in a sense that groups of people develop shared beliefs. Many beliefs are falsifiable and this what makes the discourse scientific, the falsifiability of statements [Karl Popper].
However some statements are so general that, if they are false, we are unlikely to ever find out that fact for the next 150 years. For example most statements in complexity theory are excessively general statements about the existence of certain algorithms. But existence does not mean that we can find them. They can be very large incompressible pieces of 25-th century mathematics. This kind of assumptions should not be used.
Assumptions in cryptology should be ‘efficiently falsifiable' [Moni Naor] by which I mean that if they are false, we should be able to figure it out in a short time (say less than 20 years). This means that they should not be too general and too strong, but more pragmatics and weak. The basic principle of science [known as Ockham's Razor] is to make as weak assumptions as possible. Large areas of modern

cryptology has been developed with disregard to these basic principles.

Cryptology vs. Mathematics

One Russian mathematician told us that there are problems in mathematical models being applied in the real life [cryptology]. Another mathematician told me that cryptology is a “stupid engineering” science derived from mathematics.

In fact, the view where cryptology is reduced to mathematics is rather wrong. Cryptology can be seen as an independent science, a formal theory where axioms (such as the existence of one-way functions) are added to mathematics and therefore it contains the mathematics, and not the vice-versa.

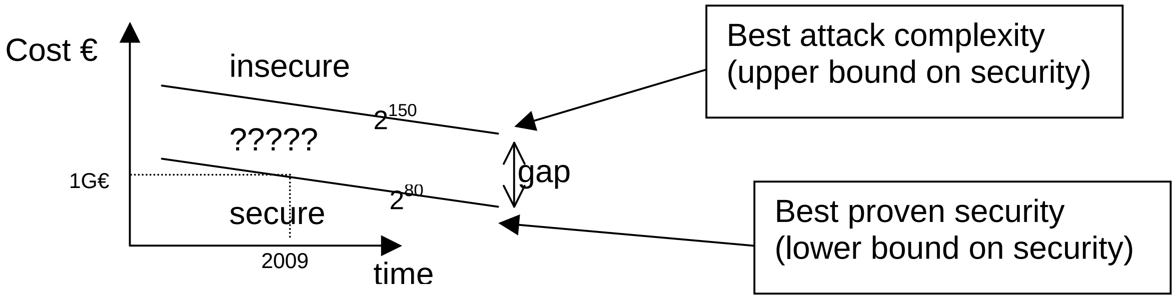
Many mathematicians hate being wrong. Great majority will refuse to take too any risks with assumptions and a new axiom will be maybe added every 100 years, and half of the scientific establishment will never ever accept it.
Cryptologists take too many risks in general but they are building practical systems. They love being wrong, it creates more jobs for cryptologists...

Cryptography is a major field of engineering with more and more practical applications. Cryptology is the science pendant to it.

Why cryptologists are paranoid? What is an Attack?

Security goals are very ambitious, and the slightest weakness will be called 'attack' and ask for the revision of the cryptographic technique in the question... This prevents future attacks.
It is only by breaking systems, that one can in the long run, know which are secure, propose better ones, and update security before it is too late.

As the science matures, the security of a system in one dimension [w.r.t to one well defined security goals 1.2.3., see below] should look like this.


Cost €
insecure
2^{150}
?????
1G€
2^{80}
secure
2009
time
gap
Best attack complexity (upper bound on security)
Best proven security (lower bound on security)

The gap can be closed (lower bound=upper bound). Then the security is fully understood. Otherwise we have a grey zone.
Many today’s industrial standards are in the grey zone.

Are cryptologists always wrong?

Quotation from Neal Koblitz [Canadian mathematician working also in cryptology]: “The Uneasy Relationship Between Mathematics and Cryptography”, In Notices of the American Mathematical Society, September 2007, see http://www.ams.org/notices/200708/tx070800972p.pdf

[…] Once I heard a speaker from NSA complain about university researchers who are cavalier about proposing untested cryptosystems. He pointed out that in the real world if your cryptography fails, you lose a million dollars or your secret agent gets killed.
In academia, if you write about a cryptosystem and then a few months later find a way to break it, you’ve got two new papers to add to your résumé![…]

Another one from [Kevin McCurley, US cryptologist with top-level maths expertise too] “The discourse regarding the role of complexity [theory] in cryptography has degenerated to a point where it may take some time to recover.”

<u>Cryptographic Research and Scientific Community:</u>
Not much is proven… and many things will never be.
A group of people with shared beliefs. Most deeply rooted in a certain reality of hardness resulting from precisely this endless confrontation of clever designers and clever attackers… Some are spectacularly naïve and are to collapse next, as usual in cryptology.

The problem is the same as before: [largely] missing markets for security technology.

Academic research, as every institution is, is prone to creating and perpetrating self-reinforcing misconceptions, suffers from bias, nepotism, conflicts of interests, hidden agenda due to being consultants to a particular industry firm/government etc...

Security and Economics

Costs and Trade-offs

Critical to any security decision is the notion of [security] trade-offs, meaning the costs – terms of money, convenience, comfort, freedoms, and so on - that inevitably attach themselves to any security system. People make security trade-offs naturally. [Bruce Schneier “Beyond Fear” 2003 book p.1].

Many costs are intangible and hard to trade for money: Paying attention, loss of freedom and privacy, being subject to unforeseen risks and consequences, etc.

Externalities

Externality: an [intended or most of the time unintended] external effect of economic activity. Like pollution, noise, etc.
Examples in the field of security: cost of benefits to unemployed workers [yes, this will be a ‘social security’ issue], some forms of crime facilitated by a firm’s activity, data security breaches, computer viruses etc.

Who Wants Security? Battles of Conflicting Interests.

Security is usually a matter of public interest…
Frequently neglected, who cares about the public interest…

To understand these questions we need to study economics and business.

What is general interest? Where a large number of people (or firms) benefit a little. But it is very hard to account for it, or make money out of it. So when one person or an organised minority interest can in a certain way benefit a lot from the opposite, even unwillingly, and even if the overall outcome for the economy is a total disaster, the negative outcome is likely to prevail.

Adoption of new technologies: Hard problem in the business circles.
Currently private companies spent more on coffee than on real security…
For example smart cards for banking have not been adopted in many countries, which result in (up to 10 times) more fraud. But some people profit from fraud.
Good points about fraud: companies keep in-house security expertise. Because they need to be reactive to the changing security landscape and keep up with competitors.
Example: one year MasterCard or Visa (I forgot which) introduced CVV. Their competitors saw that bank card fraud in this year was almost entirely borne by them. Then they copied the idea.

Excessive profits versus the interest of the economy as a whole.

Quotations from Adam Smith, Wealth of Nations, 1776, pp. 287-288 in New York: Random House, 1994 edition.

The rate of profit does not, like rent and wages, rise with the prosperity […] On the contrary […] it is always highest in the countries which are going fastest to ruin.

Excessive profits in one country have the same effect as excessive taxation, they destroy the country's economy. Capitalism works by a combination of creation and destruction, money flows into firms and later leaves them to go elsewhere.
Notable exception are the new technology firms, where very high profits are a natural reward for being innovative and very high amount of risk involved in developing any new technology. If there were no high rewards, nobody would take high risks and the technology development would be stifled.

Total lack of competition ends up becoming a lack of innovation (or privileges a wrong sort of innovation).

Incentives. Security as Externality.

Some quotations from Bruce Schneier:

- "Cybercrime [...] is not a technological problem. It's an economic problem: the incentives aren't there for smart people to solve the problem ..." [The Economist, 29 November 2003, top of the page 77].
- "lawmakers need to do more than create new punishments for wrongdoers -- they need to create tough new incentives that will effectively force financial companies to change the status quo and improve the way they protect their customers' assets. [A Real Remedy for Phishers, Wired News, October'05.]
- "As long as the banks are not responsible for financial losses from fraudulent transactions over the Internet, banks have no incentive to improve security. But if banks are held responsible for these transactions, you can bet that they won't allow such shoddy security." [Cryptogram Newsletter, 15 February 2005]

Conflicting interests, creepy corporations.

A title of a book by Bruce Schneier: "Secrets and Lies". I wonder what exactly he meant by "Lies".

Another one from Adam Smith.
[Business people; that he called 'dealers'] have generally an interest
to <u>deceive</u> and even to oppress the public [...].

Secrecy and Two-Sided Markets, Open versus Closed Platforms.

Two sided-Markets:
One company can simultaneously deal with two several groups of customers, for example Microsoft deals with individual clients and large corporations.
Companies may subsidize one activity in order to make more money form another activity. For example watch TV for free, paid by the advertisers.

Platforms:
What unites two markets will be “a platform” in which the company invests. Platforms are more or less open and more or less closed. In fact most are simultaneously open and closed depending on the point of view. It reflects strategic alliances and the necessity for any business to simultaneously compete and cooperate with other firms.

Secrecy that we study in cryptology, is an essential consideration in business. Together with legal rights these are main sources of “inimitability” and competitive advantage for business. A company cannot exist for a very long time without developing all sorts of protective barriers against competitors.

Examples of relatively open platforms:

- A PC,
- A GSM mobile phone,
- Linux,
- EMVs bank card and many 3G SIM cards,
- Programming languages such as Java or Python,

Examples of almost totally closed platforms:

- A latest Intel CPU,
- A game console,
- Windows,
- Most RFID, Pay TV and GSM smart cards,
- Proprietary crypto chips and tamper-resistant devices used in banking, the military etc.

Cryptology is one of the key technologies that allows corporations [like for major “great powers” nations in the past] to keep their secrets safe and to protect firms from theft of the technology in which they invest. If you cannot protect the investment, incentives for investment will be much weaker.

Open platforms are in general more secure? Not sure at all.
In fact secrecy and security are two totally independent dimensions [not so independent in the historical past]. More about this: see Kerckhoffs principle and the discussion around it.

Security of a PC

PC hardware used to be a totally open platform. As a result IBM didn’t make that much money but created a huge and long-lasting industry and allowed others to make lots of money. Recent tendency is to tighten/close the PC technology back again, with patents, secret specs and on-board ‘policeman’ chips such as TPM. Corporations would like to own and control each individual PC.

The PC software industry is a very special industry, unlike anything else ever seen in the history of markets. It is very rare that an industry has no legal obligation whatsoever to deliver a product that actually works and does not put people’s lives in jeopardy (cf. externalities and breaches). This industry developed with disregard to security and with contempt to the customer. The economics of the industry is based on ‘lock-in’ (high switching costs). Lowering the adoption barriers makes that customers will accept almost anything.

The quality and security of software are known to be so bad, that the software industry is sometimes presented as an example of a severe failure of the free market capitalism.

In contrast the car and aviation industry that have similar safety and security issues, have developed with an obsession with security, and are very heavily regulated until this day. These industries are low-margin industries that are necessary for the economy to function. They are also known to amplify the economical cycle, for example they are in very bad shape during a recession.

Very famous quote from Schneier: "I am regularly asked what average Internet users can do to ensure their security. My first answer is usually, "Nothing -- you're screwed."

In reality we **MUST** spend now considerable budgets to prevent our personal and financial security from being breached by hackers and criminals. We just HAVE TO pay for our security every day, if only to avoid legal responsibility for security incidents (that nevertheless will arise). But the money spent does not seem to improve the security in the long run that appears to even get worse.

Can I have my privacy/security back ?

- Yes. If you wish to pay for it, Microsoft will offer you a practical illusion of privacy and security (not on their catalogue as of yet ?).
- No. Companies that are willing to pay for your private data will buy them (and pretend they are not). The only hope: it will be expensive.

Security will be a compromise. BUT… Security is always a compromise !
Do not compromise easily. Do not sell your privacy/security for a low price.

Evolution of Security

The main tool is security; the secret (secrecy).

Three stages in information security [Courtois, Pour la Science]
and 3 degrees of evolution of cryptographic technology:

- Protections that are secret,
- Based on a secret key,
- Private-public key solutions. [Usually require tamper-resistant hardware.]

Remark 1: Many problems can be solved at each of the three levels. The third level will offer by far a superior security level.

Remark 2: Evolution is in the following direction: making secrets easier to keep and lowering the impact of their compromise…

Important: the benefits of three levels can be combined [layering the defences]: we can have a confidential public key system where the public key is NOT made public. This is frequent in the industry.

Protections that are secret

The stone age of cryptography... Like hiding the key under the doormat. Usually broken if you try long enough. Hackers paradise: just give me enough coffee…
Unpredictable and catastrophic when some information leaks out…

Based on a secret key

Shared Key.
The key remains secret.
=> Algorithm CAN be published !
Not an obligation though…

Kerckhoffs principle [1883]

“The system must remain secure should it fall in enemy hands …”

Important: It does NOT say that you must disclose the spec.
It just says that when the spec is public (which can be the first line of defence) the security should remain very good.

Computational security. Appeared with perfecting Enigma against attacks…
More and more computation, necessity to build bigger and bigger machines called “bombs”. Computational Security: time+money.
Consequences: Good Crypto ⇒ can publish the algorithm.
In 1977 the American government publishes DES. Before: good encryption algorithms were highly classified weapons.

Methods to have a good crypto algorithm:

- Prove security completely [never happened for a practical system]
- Have 100s experts and researchers actively trying to break it with large computers. Pure Darwinian (natural selection or survival of the fittest) method!
    - Either because the problem is very famous and people want to solve it for glory and fame
    - Or because there is a price. These prices are an objective measure of scientific achievement, independent from opinions of experts that are usually biased.
        - For example for breaking Elliptic Curve systems Certicom offers 725 000 $.
        - For factoring, RSA security offers 200 000 $.
        - No price for AES.

In contrast:

- Proprietary algorithms: No time, no motivation: many "lousy" algorithms, few people able to break them…How to make sure that people will try to break a system.
- Many are obvious "snake oil" systems, it shows: because what the inventors write is just pure nonsense.

Private-public key solutions

A.k.a public-key cryptography, asymmetric cryptography.
No shared key, One private and one "public" key.
Public key: Just a name no strict obligation to publish it, needs to be known be a larger number of parties, but no reason to give it to everybody.
Private key: In the past it was called secret key. Private key is a better and more modern name [private: only one person, secret can be shared].
Private key: generated and stored securely… PK cryptography usually requires tamper-resistant hardware such as smart cards. Almost unavoidable.
All these developments are ahead. Very little of it is in fact used today [still costly].

Formal Approach to Security

Modern cryptography [late 90s]

1. First: Understand what we want:
   Formal security definitions.
2. Then: Try to achieve it:
   Prove the Security w.r.t. a hard problem.

Serious Issues:
Only in about 1998 people understood what is a secure public key encryption system.
Only in about 2001 people did understood how to use RSA properly.
Most industrial standards (e.g. ISO) are not very secure or in the grey zone.
Minefield: Try-and-error just did not work…

Security vs. Safety
Difference: protect against intentional damages...
Intelligent opponent. Notion of an Attacker / Adversary.
Notion of an Attack.

Claim [Courtois, Schneier]: Computer security and real-world security are governed by the same principles.

Security: 3-point formal approach:

What is Security ? 3 key questions.
Inability to achieve sth. undesirable:

1. Security against what: Adversarial Goal.
2. Against whom: resources of the Adversary: money, human resources, computing power, memory, risk, expertise, etc..
3. Access to the system.

It is very important to understand that 2. and 3. are two totally independent dimensions.

Security Notion / Definition = a triple:

1. Adversarial Goal.
2. Resources of the Adversary.
3. Access / Attack.

One can ONLY talk about security w.r.t. a given triple. May not hold for other triple.
Examples – see my slides.

Take the STRONGEST POSSIBLE version:

1. Adversarial Goal. the weakest possible !
2. Resources of the Adversary: The strongest possible: 10 G$ budget.
3. Access / Attack: The strongest possible, total adaptive “oracle” access.

What for ? Cryptologists are paranoids. It is a part of their job.

- (Door closed ? Use the window). The philosophy: the door should be just closed, (not almost closed with 1mm space).
- It prevents against many future attacks, not only the attacks that are known today.

It is not always possible to take the strongest possible version. Partially ordered set of security notions.
Then we have to work on satisfying several security notions simultaneously.

Provable security.

Reduce the difficulty of breaking a system to a difficult mathematical (and computational) problem. All the security reduces to this single problem.

=> Does not exclude attacks, drastically reduces the number of possible attacks.

Two flavours of security proofs:

- standard model (less assumptions are needed for security, see example below).

- random oracle model - currently random oracle proofs believed to give secure and correct schemes.

Vocabulary Revision

- For messages: confidentiality==secrecy. Synonyms.
- For people: privacy: ability to control how data about you propagate, conceals properties of people, not the messages. Related notions: anonymity, pseudonymity
- Steganography (stéganos – gràfein): conceal the very existence of the "message". Message ('stego-work') is embedded in 'cover-work'.

Steganography and cryptography are two totally independent dimensions. A hidden message can be encrypted independently. Or not.

- Covert channels: use one cryptosystem and add an extra hidden channel (e.g. embed some hidden message inside a standard digital signature)
- Information hiding / embedding / watermarking: may be hidden, but may also be visible/known/readable, just make it hard to remove…

Vocabulary: Codes and Ciphers

- Confidentiality == secrecy
- code, coding theory, encode / decode, no secret here

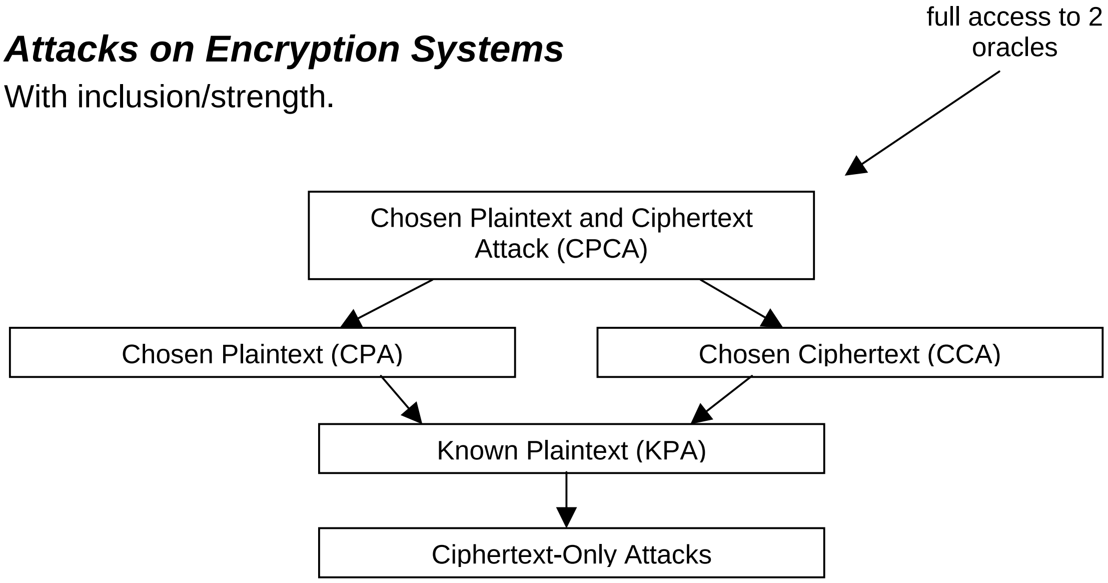
cipher = a secret code (≠historically codes were secret code-books)

- cryptosystem == cryptographic system
- clear_text == plain_text, written together or in two words
- ciphertext == cryptogram
- encryption == encipherment
- decryption >= decipherment, decryption may be without the key (breaking the code)
- cryptanalysis <= breaking the cryptosystem (in practice).

Encryption

Attacks on Encryption Systems

With inclusion/strength.



Key Sizes (for Symmetric Encryption)

What kind of keys can be cracked today:

- For a hacker running a network of PC bots: 2^{64} keys checked.
- For a large intelligence agency / excentric billionaire: 2^{80}.
  Requires custom or reconfigurable hardware devices (FPGAs or ASICs).

80-bit and 128-bit security.

- 80-bit level: Breakable, not broken yet
  (or if, they don't advertise it).
- 128-bit level: Unbreakable before we become senile. HUGE GAP !
- Fact: There is a lot of space between 2^{80} and 2^{128}.
  ¼ of million x 1 billion times more.

In optimistic scenario (Moore's Law works) 128 bit keys will be broken in 2070.
In a pessimistic scenario: will take even longer(!).

80-bit and 128-bit Security

80-bit level equivalents (nearly broken):

- 160-bits for hashing (SHA-1).
- 1024-bits for RSA.
- 160 bits for Elliptic Curves.

128-bit level equivalents:

- 256-bits for hashing (SHA-256).
- About 4096-bits for RSA (!!!!)
- 256 bits for Elliptic Curves.

RSA is open source and easy to implement.
Today RSA is roughly speaking as costly as the Elliptic curves [in terms of CPU power, ignoring the intellectual property]. As time passes by and the security requirements increase, the RSA will become much more expensive than elliptic curves. This is because the security of RSA grows VERY SLOWLY with increasing key size [sub-exponentially], while the implementation cost [cf. the carbon footprint] grows quickly [roughly as a third power] with the key size.

Key Sizes - Relevance

Question:
WinZip is using AES-256 (key size=256 bits)
Is it better than AES-128 ?

Key sizes give an upper bound on security.
Does not guarantee a lower bound.
However if the cipher is robust enough, they will NEVER be an attack much faster than 2^n.
Key Sizes - relevance
AES-128: broken faster than exhaustive search,
may mean broken in year 2070…

Therefore there is no probably no point using 256-bit keys, except for e.g. military secrets where a life span of 120 years can just be mandatory requirement (don't discuss it, just do it…).

Key Sizes
Is AES-256 better than of AES-128. ?

The opposite can be argued: (imagined scenario)
AES-128 may never be broken in less than 2^{128}.
AES-256: somebody will publish an attack in 2^{253}. He will become famous for breaking AES, yet nothing will happen. In year 2200 AES-256 will still remain unbreakable.

Besides some people believe that 1) quantum computers cannot work 2) the total number of atoms in the accessible part of the universe is less than 2^{256}.
Then the key of 256 bits will never be cracked. Not even in 10 000 years.

Probabilistic Encryption

There is a difference between cryptographic primitives (block ciphers, stream ciphers) that are deterministic algorithms with limited usage and limited security (a block cipher encrypts only short blocks), and a full-fledged encryption system that can be used in practice (encrypt arbitrary data), and should be probabilistic.

Randomness is a very important ingredient that will increase and enhance the security in many different ways. Any “good” practical encryption system should be probabilistic. Main reasons are:

The same message is encrypted in many different ways. Implies redundancy.

- Makes cryptanalysis harder (harder to combine knowledge gained).
- Prevents the enemy from realizing when the same message is sent twice.
- Prevents replay of an authentic message.
- Allows tracing of messages (!).
- Allows to check the authenticity of the message (in some cases it can be proven correct).

Nearly all secure modern cryptographic techniques require randomisation and are NOT secure without.

One Time Pad and Unconditional Security

Perfect security (for encryption we have prefect secrecy or perfect confidentiality)
== secure against unbounded adversaries
== secure against an enemy that has so called “infinite computing power”
== secure in information-theory sense

Aspect of one-time pad (OTP a.k.a. Vernam cipher).
Very important:

Theorem. The Vernam cipher achieves perfect secrecy FOR EVERY DISTRIBUTION of the plaintext M. (but the key should be random and uniform).

Not just when the plaintext is random and uniformly distributed. Very important: whatever is the A PRIORI KNOWLEDGE we have on the plaintext, by looking at the ciphertext we cannot learn more.

BTW. This property also exists in computational security (as opposed to unconditionally secure OTP): and is very important in public key cryptology; it is called semantic security [a.k.a. Indistinguishability IND-…]

The Link between One-time Pad and Semantic Security

Perfect Secrecy: - definition -
The a posteriori distribution of the plaintext M =
the a priori distribution:

$$\mathbf{Pr\ [M=m|\ C=c] = Pr\ [M=m]}.$$

Perfect Secrecy: - an equivalent definition [J.Katz]:
For any two messages m_1 and m_2, and for any algorithm A we have:

$$\Pr[A(C)=m_1 | C = E_K(m_1)] = \Pr[A(C)=m_1 | C = E_K(m_2)]$$

(Remark: there is no limitation on the speed of this algorithm
== the Adversary is unbounded.)

Semantic Security: - computational version [incomplete/informal
For any two messages m_1 and m_2, and for any efficient algorithm A we have:

$$|\Pr[A(C)=m_1 | C = E_K(m_1)] - \Pr[A(C)=m_1 | C = E_K(m_2)] < \varepsilon$$

Remark: this (still informal) IND definition applies to both public and secret key encryption (!).

Difference: in PK the access to encryption oracle is always granted by the public key.

Semantic Security [Goldwasser, Micali, 1984, for PK setting]
Other Names:

- Indistinguishability,
- IND, (two major versions: IND-CPA and IND-CCA)
- Polynomial security,
- Strong secrecy [a more general term]
- [Left or Right Indistinguishability]
  – proven equivalent in (at least) some cases.]

Shannon Would Love It !
Exact equivalent of perfect secrecy in the computational setting.
In fact more: the system can be used many times…

Fact:
A deterministic (and stateless) encryption system cannot be semantically secure w.r.t. known plaintext (and any more powerful) attack. Must be probabilistic.

(Semantic Security – what is missing to have a fully formal definition ?
We need to specify "access/attack/3.", such as two oracles in IND-CP2-CC2 etc.)

Key Management

Very complex question, depends on many physical/system and human factors.

Main methods for deriving cryptographic keys.

- All key are identical [and equal to default key], no kidding, this is happens 10 times per day in the industry as discovered by hackers… Example Diebold voting machines scandal in the US.
- Session key derivation: one example is GSM, another is MiFare Classic (London Oyster card). The key used for encryption is basically derived from the key established during a [mutual-not in GSM] authentication protocol.
- Key diversification: use symmetric cryptography (for example a hash function or more generally a PRF) to derive key from the number/ID of each person/device. Very common practice in the industry (e.g. GSM AuC). Can lead to a very major compromise (real-life example: Chrostopher Paar paper about KeeLoq at Crypto – or how to break to EVERY car of the same manufacturer with a master key).
- Public key crypto: each device generates its own key and ONLY public keys are transported, authenticity alone is sufficient for security of the system.
- Identity-based and certificated-based crypto: promises less infrastructure.
    - The public key is derived from one's name, email address or other unique ID.
    - Then the authority gives you your private key.
        - One can send a message to the person that never subscribed to any system.

Key life cycle, transport and security of keys: major headache.
Keys need to be:

- Securely generated,
- Securely transported,
- Securely backed-up,
- Securely destroyed/erased,
- Securely transmitted,
- Securely used (see side-channel attacks),
- Certified through a 'secured' process – see qualified digital signatures, and PKI.

Kleptography: embedding a backdoor that leaks keys to the attacker.

The n^2 problem.

One key per each pair of people? Public key cryptography is the only scalable solution for large open networks.

Key Generation

Vocabulary

1. “Real”/Physical Random Number generator:
   - Comes from fundamental physics, complexity (imprecision of measurement) or intrinsic randomness (quantum mechanics).
2. Pseudo-random number generator:
   - Expands a short random string into an infinite sequence of random-looking bits.

Keys vs. Bits
Symmetric Cryptography:
Key should be just random, e.g. 128 bits.
Hacker remark: But for many encryption modes, when in memory, to save about 2/3 of running time, keys are stored expanded and are easy to recognise…

Asymmetric Cryptography:
Keys have some algebraic structure,
Complex setup.
Some parameters can be system-wide (for several users).
Other are private. May still be hard to generate.

Hard or Easy ?

Q1. Is it easy to generate “secure” random numbers ?
Yes. Many simple methods work.

Q2. Is it easy to distinguish the best possible source of randomness from a fake one with a hidden setup/trapdoor ? Impossible in 99.9999 % of cases.

Another question:
Assume that we have a “Real”/Physical Random Number generator + some treatment (to remove imperfections or bias). Should it be used ?

Answer: never. Not secure enough.
Source: many people (e.g. the French government DCSSI in their recommendations for the industry) explicitly rules out this usage.

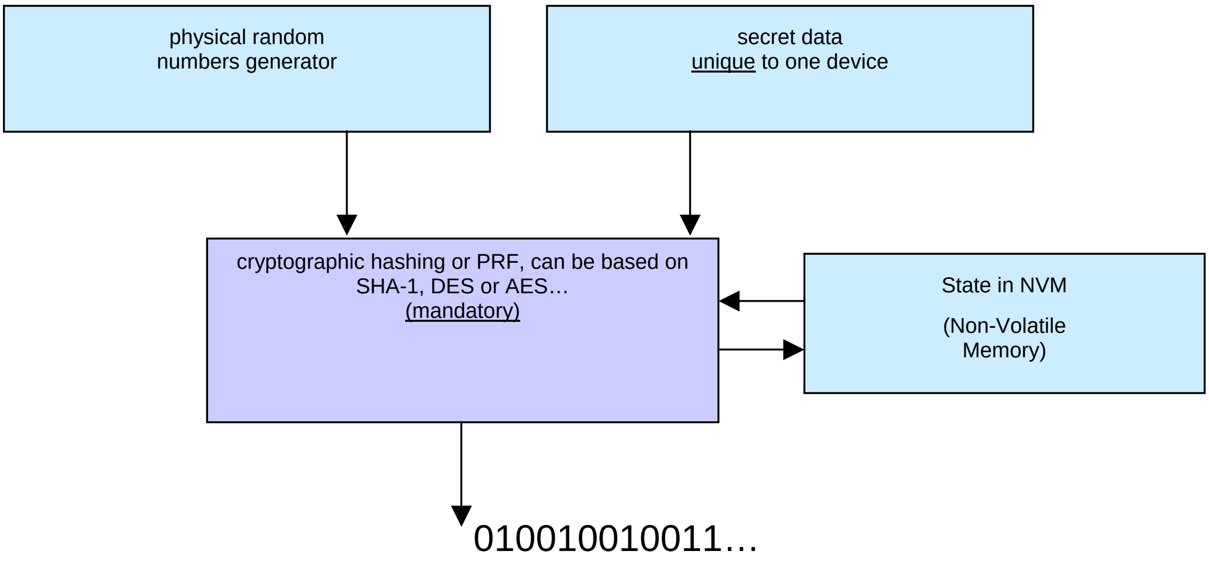
Reason: can be broken or be disabled by the attacker.

Entropy Extractors

Idea: Use hash function.
Any, even MD5 will do wonderful job. Does not have to be CR…

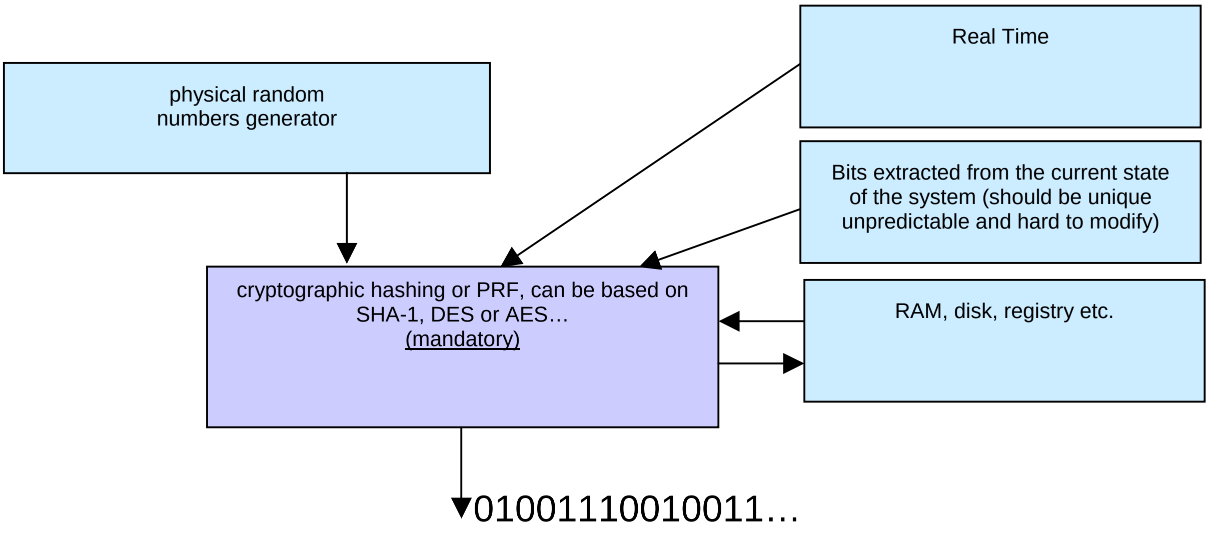
Claim: If there is some randomness in the initial generator, the hash function is sure to find it.

Entropy Extractor Paradigm – like hashing, better.
Add NVM, prevents reset attacks.
Finally, recommended architecture:


physical random numbers generator
secret data unique to one device
cryptographic hashing or PRF, can be based on SHA-1, DES or AES… (mandatory)
State in NVM
(Non-Volatile Memory)
010010010011…

Each device should produce unique numbers.

What about software?


Real Time
physical random numbers generator
Bits extracted from the current state of the system (should be unique unpredictable and hard to modify)
cryptographic hashing or PRF, can be based on SHA-1, DES or AES… (mandatory)
RAM, disk, registry etc.
01001110010011…

Pseudo-Random Number Generator

Definition: Expands a short random string into an infinite sequence of “random-looking” bits. Can be a stream cipher.

Usage:

- where there is no / poor sources of randomness available.
  Real randomness is in fact expensive (e.g. slow).
- Where we must be able to reproduce the computation exactly (deterministic randomness).

Formal definitions: skip, similar as for OWF.
Inability to distinguish from a really random string of bits.

Pseudo-Random Number Generator
How to Make One ?:

- Method 1: Use a synchronous stream cipher such as Snow 2.0.
- Method 2: use a block cipher or hash function. Just iterate starting from some initial state. State size >= 160 bits. Still very good if 128 bits and if the number of bits used is much less than 264.

Public Key Cryptography

RSA Key Generation

Requires to generate 2 large prime numbers. E.g. 1024 / 2 bits each.
Not an easy task, and slow. Many smart cards needs tens of seconds for this. Not so serious because done only once in their lifetime.

For example, on Infineon futuristic 32-bit 0.13 um chip SLE88CFX4002P, it takes 3.5 seconds on average to generate a 2048-bit key with the maximum clock speed of 66 MHz. According to Infineon, SLE88CFX4002P is "the most sophisticated smart card microcontroller" on the market.

Random Primes
Method:

- Choose a Number at random.
- Check if it is a prime. (isprime(n) in Maple).
- Try again…

This gives you a guarantee of prime numbers that have no bias or special structure.
Requires many tries…
Theorem: The probability that n is a prime is about 1/log n.

Remark: Can do much less tries by “sieving”:
one only needs to check integers that are congruent to 012345 mod 6. Etc.

Random Primes
Question: [important because randomness is an expensive resource (!)]:
How much random bits do we need to generate a random prime on 1024 bits ?

Answer: about 1024 bits only.
Method:
Choose a random number x on 1024 bits.
Pick the smallest prime p >= x.

Primality Testing
In 2002 Agrawal, Kayal and Saxena [AKS] have shown that Prime is in P.
Means to check if a number is a prime takes polynomial time.
Based on Fermat's Little Theorem.

In practice Miller-Rabin is faster.
Many many other methods - see http://cr.yp.to/primetests.html

Miller-Rabin is polynomial in practical (not theoretical) sense.
What's Wrong with These Methods ? Conceptual Problem:

- They either do not guarantee that the number is 100 % prime (it will be a prime with probability $1 - 2^{-80}$).
- Or they are not guaranteed to terminate in polynomial time.
- Or both…

Perfectly good for cryptographic applications.

DH/ElGamal Key Generation

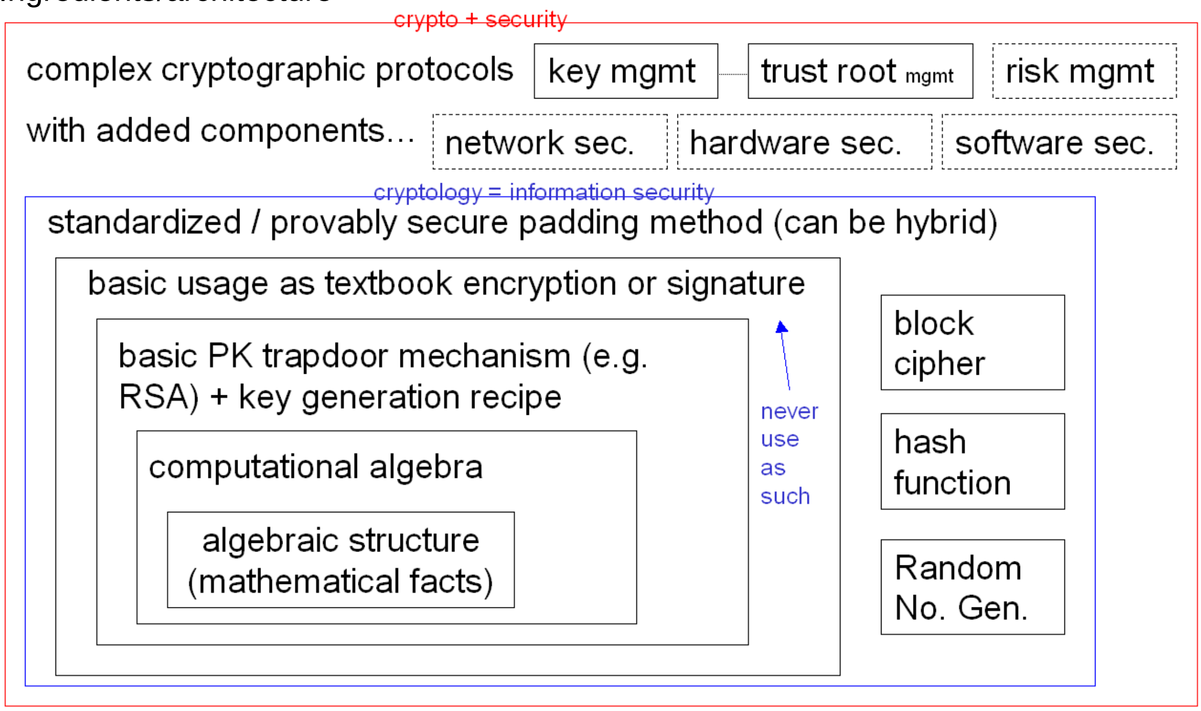
ElGamal and DH Key Generation in Zp*
Setup: g, a generator of Zp*.
p prime >= 1024 bits.

How to find a generator ? NO EASY METHOD KNOWN (!).

- Method 1: Use a prime with p=2p'+1 or similar.
- Method 2: Requires to factor p-1. Fact [by Lagrange Thm]: if for all prime p_i | p-1 we have g^(p-1)/pi <> 1 then g is a generator.

Towards Secure Public Key Cryptography

Ingredients/architecture



crypto + security

complex cryptographic protocols

key mgmt

trust root mgmt

risk mgmt

with added components…

network sec.

hardware sec.

software sec.

cryptology = information security

standardized / provably secure padding method (can be hybrid)

basic usage as textbook encryption or signature

basic PK trapdoor mechanism (e.g. RSA) + key generation recipe

computational algebra

algebraic structure (mathematical facts)

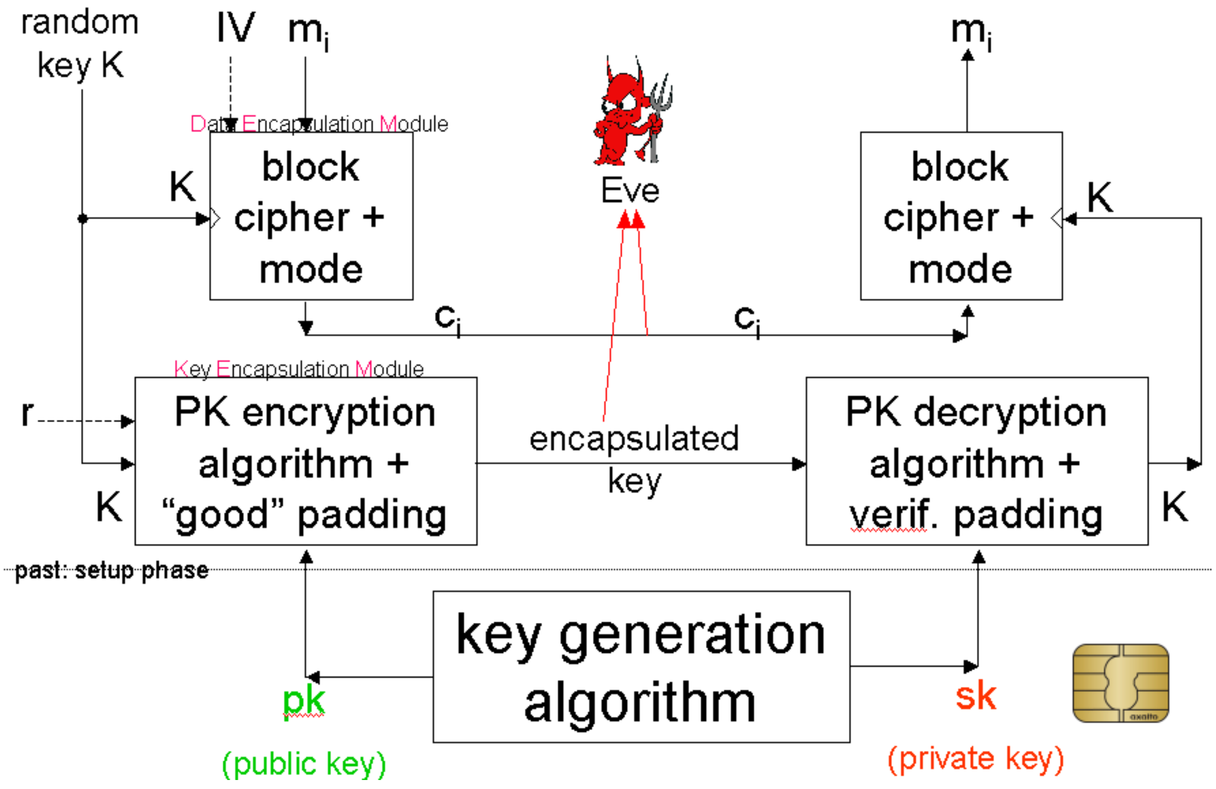
never use as such

block cipher

hash function

Random No. Gen.

Hybrid Encryption (PK is by far to slow!!)



random key K

IV

m_i

Data Encapsulation Module

K

block cipher + mode

c_i

Eve

c_i

m_i

block cipher + mode

K

Key Encapsulation Module

r

PK encryption algorithm + “good” padding

K

encapsulated key

PK decryption algorithm + verif. padding

K

past: setup phase

key generation algorithm

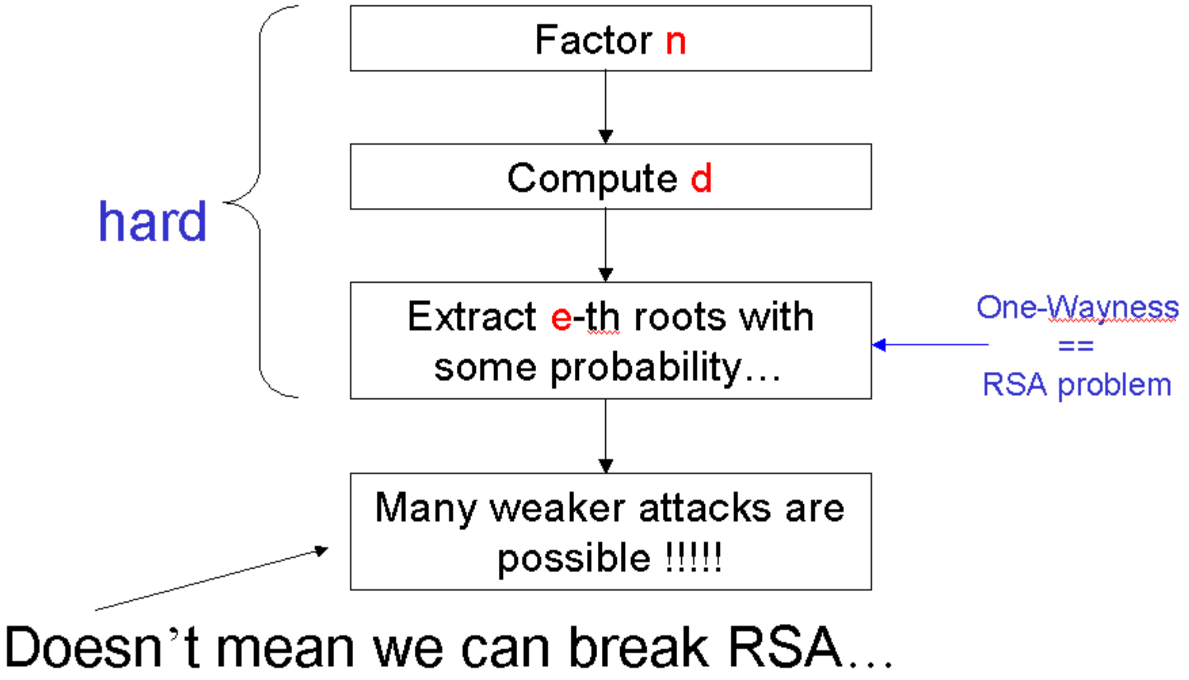
pk

(public key)

sk

(private key)

Security of RSA (revision)


hard

Factor n

Compute d

Extract e-th roots with some probability…

One-Wayness == RSA problem

Many weaker attacks are possible !!!!!

Doesn’t mean we can break RSA…

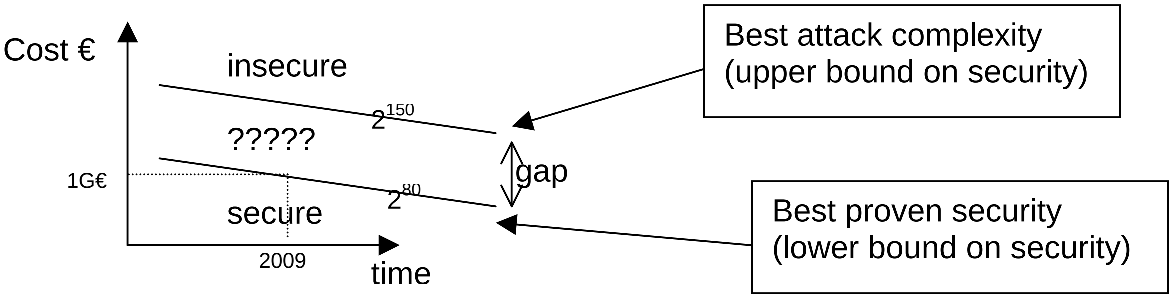
Text-book RSA is secure (basically) only when it is used a limited number of times to encrypt different random-looking messages (it is OW-CPA).
Provably secure padding: allows to transform a weakly secure = > strongly secure (IND-CCA2) system.

Public Key Cryptography vs. Provable Security.

Including a discussion of a number of things outside the scope of COMPGA12 but that you may hear of in Crypto 2 course, your research or professional life.

Not vague notions - VERY PRECISE ! Not theory, practice !
This is the area where the science was VERY successful to bridge the theory and practice. We have “a practical theory”.
A number of problems are solved in an elegant provable secure way. In symmetric cryptography they usually aren’t anywhere near.

Pay attention to details: "Provable security" – maybe means not secure enough ? E.g. RSA-OAEP.
"Practical Security" = tight security reductions = no gap ! = cannot do better !


Cost €
insecure
2^{150}
?????
1G€
2^{80}
gap
secure
2009
time
Best attack complexity (upper bound on security)
Best proven security (lower bound on security)

Basic Security Notions (revision*)*

OW-CPA: insufficient security:

- secure if used once in your lifetime to encrypt a completely random message…

IND-CPA: minimum security, “strong secrecy”:

- given the public key + ANY context information about the message, cannot decrypt !
- Perfectly good in practice if "insider attacks" are excluded: if there is no virus on the PC or terminal to which the smart card is connected, no spy (military), no dishonest employee (banking) etc.

IND-CCA2: maximal security:

- secure against all (known+future) attacks including insider attacks.

Encryption (harder, not solved by current international standards):

- ElGamal: OW-CPA under CDH and IND-CPA under DDH.
  - o "Minimum security": when no "insider" attacks, no viruses, etc.
- OAEP: not secure in general, secure for RSA, not tight security:
  - o OK for >4096 bits, but 16384 bits needed for military-grade security..
- OAEP+: secure in general, still not tight.
- OAEP^{++} and OAEP++: two variants with a tight security proof.
  - o Not yet published. Future standard of RSA security ?
- Boneh-Rabin simplified OAEP, advocated by Koblitz and Menezes in eprint/2006/152/, tight security proof and the only one based on factoring !
- RSA-BR: Generic construction of Bellare-Rogaway'93: Perfectly OK, tight sec proof (cf. Pointcheval DEA slides).
  - o A fully secure method to encrypt with RSA >=1024 bits.

Signature (easier):

- RSA-PKCS #1 v1.5. insecure (no proof yet, not broken, variants broken)
  - o (exists also in PKCS #1 v2.0 and 2.1 cf. www.rsasecurity.com)
- RSA-FDH: perfectly OK. Except how to find hash function on 2048 bits ?
  - o BEWARE: hash function may be broken.
- RSA-PSS: current recommended standard, part of PKCS #1 V.2.x.
  - o The best method to sign with RSA >=1204 bits

    (except BEWARE: hash function may be broken…)

What Can be Achieved? Examples

Two flavours of security proofs:

- standard model (less assumptions are needed for security, see example below).
- random oracle model - currently random oracle proofs believed to give secure and correct schemes.

What can we achieve exactly ? Two examples:

1. El-Gamal is IND-CPA (minimal security, OK for many applications) if:
   - The security proof is correct and tight – OK.
   - The CDH problem is hard in G=<g>. This implies in general that:
     - o q = Order(g) is at least 160 bits.
     - o If $G=Z_p$* then p is at least 1024 bits.

Nothing more needed – standard model security is achieved.

2. Both RSA-PSS and RSA-FDH are secure (EF-CMA: the strongest level) if:

- The security proof is correct and tight – OK.
- The Random Oracle assumption is correct: means roughly that every attacker is behaving as if the hash functions used were a black box…
- The RSA problem is hard. This implies that N is at least 1024 bits.

Nothing more needed.

More over, a practical deployment of the above schemes is only secure if:

- Your random r is really random (example: a smart card without a good random number generator is <u>never</u> secure).
- The public key is authentic (not easy to achieve in practice, requires several assumptions, see below).

PKI

Public Key Solutions – deployment in real life. Public Key Infrastructure (PKI):

- CA = Certification Authorities = A network of publicly known trusted authorities that sign other people's keys +
- Employees that will verify the identity of the people and deliver them badges, and sign their public keys (US Army uses this with Axalto cards). Cost of this usually 10 times more the cost of actual smart cards (!).

Fundamental problem: asymmetric cryptography is only possible if the public key is authentic (!). This can be achieved by either:

1. An authentic key is obtained from a trusted source, was not modified by the enemy, and the program that makes verifications using this key was not modified. This assumption <u>cannot be avoided in any system</u>, i.e. at least one authentic key and at least one trusted verification program should be available.
2. The authenticity of many other keys can then (and only then) be assured by digital signatures. This can be done in two ways:
    - A <u>hierarchical</u> way, so that we need to have a chain from the root certificate to the certificate of our key.
        - Advantage: Remark that only the root certificate needs to be known in advance. All the other can be transmitted with the message.
        - Problem: if the root key is broken or revoked, the whole system is compromised. Very dangerous !!!
    - <u>Web of trust solutions</u> (PGP): any user can sign the key of any user.
        - Advantage: less expensive, ad-hoc, no Big Brother connection.
        - Serious issues:
            - if the trust is transitive, the security is very weak: one dishonest user can introduce millions of false keys to the system.
            - If the trust is not transitive, then in practice it will be very difficult to get a trusted key for our collaborators (the network of trust relationships will not be "dense enough").

CRL = Certificate Revocation Lists. When I verify a certificate, I should connect to a web server and check if some of the keys were not revoked (broken or compromised). Problem: I need to be sure that one of the keys is trusted in order to verify the authenticity of the CRL.

Another major problem with hierarchical PKIs: it is sufficient to compromise one out of many branches of the certification tree. One can create a certification authority for the lone purpose of selling real-false certificates (!). Thus, a certification authority may only decide that only keys at distance 1 from the root are valid. This means that you have to pay a lot for each key (nobody does it for free and needs to make verifications on your identity).

Other security issues with PKI:

- Homonymy – which John Robinson is he ? She claims she is married and before her name was.. but is it true ?
- To what extent the CA itself knew what is written inside this certificate ? What checks did they perform ?
- Is the user part of the security design ? With current shopping on the internet the user can decide if he trusts or not this particular site… Can he choose not to trust a particular crypto algorithm (e.g. MD5 is broken) etc?

Tips: How to design a failure-proof trust structure of a PKI?
It is possible with redundancy:

- If each key is signed several times by several CAs/users, and for example we accept one user public key if it is signed by a CA and by somebody we know personally. When sending a message the user should be informed of how "trusted" is the key (e.g. signed by 3 different people and the CA – very good) and the user should decide whether he trusts it or not – depends also how important this particular email is etc...
- Also use bio-diversity: sign with several algorithms: one signature with RSA-PSS, one with ECDSA, one with McEliece. Verify all certificates and all CRLs.

Economic Aspects of PKI

What? Do we have to pay 700 $ /year for one signature? It costs nothing to compute. YES, we must pay. Could be cheaper though.

A signature costs nothing to compute. BUT it is expensive because of the cost of:

- individual enrolment of users in a PKI system,
- verification of the identity of people who’s keys we sign,
- protect the PKI infrastructure against cyber-attacks,
- maintain revocation capabilities, need to update CRL frequently, run massive servers, make sure other people do the checks

- employ guards to guard the premises,
- protect data against fire, war and terrorist attacks etc.
- audit costs,
- criminal threat: high-level public keys are a high profile target that would allow criminals astronomical benefits. Needs very heavy protection, like a vault in a major Swiss bank, may cost millions of dollars to protect it.
- use very expensive military-grade 'tempest' hardware to protect the private keys against corrupted employees (insider threat, e.g. a Verisign employee),
- legal obligations: see EU directive requirements for "qualified digital signatures". These requirements imply many costs.
- economics and business reasons: network effects and winner-takes-it-all situation: If the market is too fragmented, it will fail, because there will be too many competing systems and no one will be practical to use (lack of other people that would use the same platform as me). So we need a monopoly (or very limited choice of major competing platforms). Which means that prices may go up in an unreasonable way [monopoly or oligopoly], yet the competition will not be able to challenge the monopoly.

Arguments against paying for it:
As a consequence of high prices, half of web sites have an invalid certificate. So we click on this box, "proceed anyway". We get used to that.
And we click on this box again when the hacker is playing the man in the middle.

Third Way: So it costs money, but people don't want to pay.
PKI is a critical infrastructure for the digital economy. So maybe the is solution is that government should pay for these things, or sponsor critical parts of this infrastructure to decrease the costs? Like the government always paid the cost of printing [must be very secure] bank notes, issuing [again must be very secure] passports and ID cards etc. And in specific cases it delivers ID cards and passports for free. Etc.