Software Security: Squaring the Circle?

Bart Preneel
COSIC – KU Leuven – Belgium
Firstname.Lastname(at)esat.kuleuven.be
http://homes.esat.kuleuven.be/~preneel
May 2015

Security goals

- Entity/device authentication
- Secure memory
- Secure update
- Data authentication (e.g. payment transactions)
- Secure boot
- Copy protection
- License enforcement
- Secure execution
- Protecting trade secrets

Outline

- Crypto
- Software versus hardware security
- The road ahead

Cryptography

- Cryptography has been very successful in terms of scientific methodology
  - elegantly solving a narrow problem
- Moving problems to keys and mathematical assumptions

Disclaimer: security ≠ cryptography

Most systems break elsewhere
Cryptography is more bypassed than broken
But if crypto is broken, there is trouble

Crypto history

- pre-1915: manual encryption or simple devices
- 1915: rotor machines: (electro-)mechanical
- 1960’s: electronic encryption
- 1975: integrated hardware
- 1990: software
- 2015: everywhere
Deployment of cryptography

- most crypto in volume and market serves for data and entity authentication
  - code updates
  - payments: credit/debit/ATM/POS and SSL/TLS
- confidentiality
  - government/military secrets
  - DRM/content protection
  - telco: not end-to-end or with a backdoor
  - ehealth (growing market)
  - most data in the cloud is not encrypted

Symmetric Key Deployments ~19B

<table>
<thead>
<tr>
<th>Technology</th>
<th>Market Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mobile</td>
<td>6.3B</td>
</tr>
<tr>
<td>Access</td>
<td>8B</td>
</tr>
<tr>
<td>Banking</td>
<td>1.5B</td>
</tr>
<tr>
<td>Blu-ray/DVD</td>
<td>500M</td>
</tr>
<tr>
<td>Hard disk</td>
<td>300M</td>
</tr>
<tr>
<td>Pay TV</td>
<td>250M</td>
</tr>
<tr>
<td>Game consoles/access readers</td>
<td>200M</td>
</tr>
</tbody>
</table>

Public Key Deployments ~9B

<table>
<thead>
<tr>
<th>Technology</th>
<th>Market Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>3B</td>
</tr>
<tr>
<td>Browser</td>
<td>2.7B</td>
</tr>
<tr>
<td>HTTP over SSL</td>
<td>2B</td>
</tr>
<tr>
<td>DNSSEC</td>
<td>600M</td>
</tr>
<tr>
<td>SSH</td>
<td>500M</td>
</tr>
<tr>
<td>SSL</td>
<td>3.5B</td>
</tr>
<tr>
<td>Telnet</td>
<td>200M</td>
</tr>
<tr>
<td>Telnet/SSH</td>
<td>37M</td>
</tr>
<tr>
<td>Telnet/TCP</td>
<td>10M</td>
</tr>
<tr>
<td>Telnet/UDP</td>
<td>1M</td>
</tr>
<tr>
<td>Secure SSL</td>
<td>500M</td>
</tr>
<tr>
<td>Non-Invasive</td>
<td>29000M</td>
</tr>
</tbody>
</table>

Side-Channel Leakage

- Physical attacks ≠ Cryptanalysis
  - (gray box, physics)
  - (black box, maths)
- Does not tackle the algorithm’s mathematical security

Classification of Physical Attacks

- Active versus passive
  - active: perturbate and conclude
  - passive: observe and infer
- Invasive versus non-invasive
  - invasive: open package and contact chip
  - semi-invasive: open package, no contact
  - non-invasive: no modification

Cryptography

- Needs next to mathematical also physical assumptions
  - authenticated copies of public keys
  - keep secret and private keys secret
  - guarantee correct execution of algorithm
  - limited leakage of internal variables
Cryptography in software

In software implementations the attacker typically has full access to the device: whitebox attack model.

Easy to find the key [Shamir-van Someren’98]

Whitebox cryptography

Implement block cipher in such a way that it is hard to extract the key [Chow-Eisen-Johnson-van Oorschot’02]

Whitebox cryptography (2)

• Attacker can always evaluate the function
  – need input and output encodings: problem is shifted to secrets in other places

• Limitations: copy the code to another device
• Other idea: insert code to be executed into the look-up tables

Practical whitebox constructions

• Several attempts in the literature (DES, AES) between 2002 and 2010
• By early 2014 they were all broken

Several companies still have proprietary designs

Does whitebox cryptography yield an efficient public-key encryption scheme?

Answer: no ☹

• table lookups can be reversed layer by layer
But strong obfuscation would yield such a scheme [Hellman, ca 75]

Code Obfuscation*: Goal

Make programs “unintelligible” while maintaining their functionality
– example from Wikipedia:

http://people.csail.mit.edu/shaih/pubs/IndistinguishabilityObfuscation.pptx
Software security
Bart Preneel
May 2015

Code Obfuscation: Why

- Hiding cryptographic keys
- Hiding license checks
- Prevent cheating in online games
- Hiding nature of 0-days that are being patched
- Hiding algorithm
  - example: assume you have a winning strategy for chess

Code Obfuscation

- Widely used in practice: art or science?
  - CFG flattening, virtual machines, data obfuscation,…
  - some rules-of-thumb, sporadic tool support
  - Wikipedia: “At best, obfuscation merely makes it time-consuming, but not impossible, to reverse engineer a program”
- Can this be made scientific?

Defining Obfuscation

- An efficient public procedure \( O(\ast) \)
  - everything is known about it
  - except the random coins that it uses
- Takes as input a program \( C \)
  - e.g., encoded as a circuit
- Produces as output another program \( C^\prime \)
  - \( C^\prime \) computes the same function as \( C \)
  - \( C^\prime \) at most polynomially larger than \( C \)
  - \( C \) is “unintelligible”: defining this is tricky

What’s “Unintelligible”? Attempt 1

- Ideally: cannot do much more with \( C \) than running it on various inputs
- Virtual Black Box Obfuscation (VBB): anything that can be efficiently computed from \( C \) can be efficiently computed given oracle access to \( C \)

What’s “Unintelligible”? Attempt 1

- VBB: If \( C \) depends on some secrets that are not readily apparent in its I/O, then \( C^\prime \) does not reveal these secrets
- generic VBB is impossible [Barak+01]
  - Thm: If PRFs exist, then there exists PRF families \( F = (f_a) \), for which it is possible to recover \( s \) from any circuit that computes \( f_a \).
  - these PRFs are unobfuscatable

Unintelligible: Virtual Black Box

For a few functions, VBB obfuscation is feasible E.g. point-function/cryptographic locks

\[
f_{a,b}(x) = \begin{cases} 1 & \text{if } x = a \\ 0 & \text{otherwise} \end{cases}
\]

Even if there are extensions, it is clear that we need to weaken the requirements if we want a universal obfuscator (i.e. works for every function)
What’s “Unintelligible”? Attempt 2
Indistinguishability Obfuscation (iO)

- If $C_1$, $C_2$ compute the same function (and $|C_1| = |C_2|$) then $O(C_1) \approx O(C_2)$ [Barak+01];
  - indistinguishable even if you know $C_1$, $C_2$
- Note: inefficient iO is always possible
  - $O(C) =$ lexicographically 1st circuit computing the same function as $C$
  - (canonical form)
- Canonicalization is inefficient (unless $P=NP$)

Best-Possible Obfuscation [Goldwasser-Rothblum’07]
Efficient iO $\Rightarrow$ hides as much about the input circuit as any circuit of a given size

Many Applications of iO

- AES $\Rightarrow$ public key encryption [GGH+13, Sahai-Waters’14]
- Witness encryption: encrypt so only someone with proof of Riemann Hypothesis can decrypt [Garg-Gentry-Sahai-Waters’13]
- Deniable encryption [Sahai-Waters’14]
- Functional encryption: noninteractive access control [GGH+13], $\text{Dec} (\text{Key}_y, \text{Enc}(x)) \Rightarrow F(x, y)$

Constructions of iO

- [GGH+13]= [Garg-Gentry-Halevi-Sahai-Raikova-Waters] FOCS 2013
  - iO for NC¹ (polynomial-size, log-depth circuits) using multi-linear jigsaw problem
  - iO for general functions: NC¹ construction + Fully Homomorphic Encryption
- Follow-up work to streamline assumptions, e.g. [Pass-Seth-Telang’14]

Practical? Constructions of iO

- 2-bit multiplier
  - 4 inputs bits
  - 1-8 AND gates per input bit
  - Generating one obfuscation takes $10^{27}$ years on a 2.6 GHz PC
- Implementation in memory: 20 Zetabyte ($10^{21}$)
- Circuit evaluation: 1.8 .10⁸ years

Number of obfuscation papers on IACR eprint
Limitations of iO

- No guarantees on leakages of information
- No black box construction of a collision resistant hash function with a polynomial security loss from a general purpose iO and a one-way (or trapdoor) permutation [Asharov-Segev'15]

- Note that obfuscation only intends to resist passive attackers

Obfuscation $\approx$ Homomorphic Encryption

$F$ $\approx$ Obfuscation $\xrightarrow{+}$ $F(x) \approx$ Result in the clear

$F$ $\approx$ Encryption $\xrightarrow{+}$ $F(x) \approx$ Result encrypted

FHE = infeasible but somewhat FHE feasible today

Obfuscation $\approx$ Multiparty Computation

$F$ $\approx$ Obfuscation $\xrightarrow{+}$ $F(x)$ $\approx$ Result in the clear

$F$ $\approx$ Encryption $\xrightarrow{+}$ $F(x)$ $\approx$ Result encrypted

Software security

"it's turtles all the way down"

preacher Joseph Frederick Berg (1854):

- My opponent's reasoning reminds me of the heathen, who, being asked on what the world stood, replied, "On a tortoise." But on what does the tortoise stand? "On another tortoise." With Mr. Barker, too, there are tortoises all the way down.

Need for hardware security

- Integrity protected storage for hash value or public key
- Secure key storage
- AES instruction
- RNG
- TPM or smart card (SIM)
- HSM

Trustworthy subsystems that can be certified
Secure hardware pitfall

- It is great to have secure hardware.
- But how do you authenticate who/what can talk to it?

Symmetric cryptography:

- Block ciphers, hash functions, etc.

  - **Design:** Easy
  - **Attack models:** Clear
  - **Security evaluation:** Complex
  - **Performance evaluation:** Reproducible
tools
  - **Tools for security evaluation:** Few are shared

Side channel attacks: DPA, CPA, Higher Order

- **DPA, Mutual Information Analysis**

  - **Design:** Hard
  - **Attack models:** Disputed
  - **Security evaluation:** Very complex
time consuming
  - **Performance evaluation:** Platform dependent
  - **Tools for security evaluation:** Some developments
  - **DPA competition:** Moderate success
  - **Industry switched to security by obscurity**
  - **Academia:** Toy implementations

Software security: obfuscation

- **Design:** Very hard
- **Attack models:** +/- clear
- **Security evaluation:** Manual and ad hoc
  most schemes easy
to break
- **Performance evaluation:** Platform dependent
  some tools
- **Tools for security evaluation:** Few
- **Industry still in security by obscurity model**
Challenges (1): industry

- security by obscurity: is this scientific? (August Kerckhoffs)
  - risk of backdoors
- commitment to help bridging the gap
- need to think about certification
  - in collaboration with academia
  - which information about the certification is public?
  - how is information shared over the chain from vendor to customer to end user?
  - what about backdoors?

Challenges (2): academia

1. Metrics
2. Metrics
3. Metrics

Scientific approach
- Measurable
- Reproducible
- Tools

Cannot burn many person-months of graduate students on reverse engineering

Challenges (3): hackers

Strive for security based on mathematics that is hard to bypass

[Bill Neugent]
Most cryptosystems are secure because most people would rather eat liver than do mathematics

Challenges (4): collaboration

- industry: explore collaboration and sharing methods
- academia: evaluate realistic implementations with combined countermeasures
- develop joint solutions, tools and evaluation methods

The end

Thank you for your attention