Advanced Software Protections

Paolo Falcarin

Ca' Foscari University of Venice

Department of Enviromental Sciences, Informatics and Statistics

paolo.falcarin@unive.it

CM0626 – Software Security

CM0631-2 – Software Security

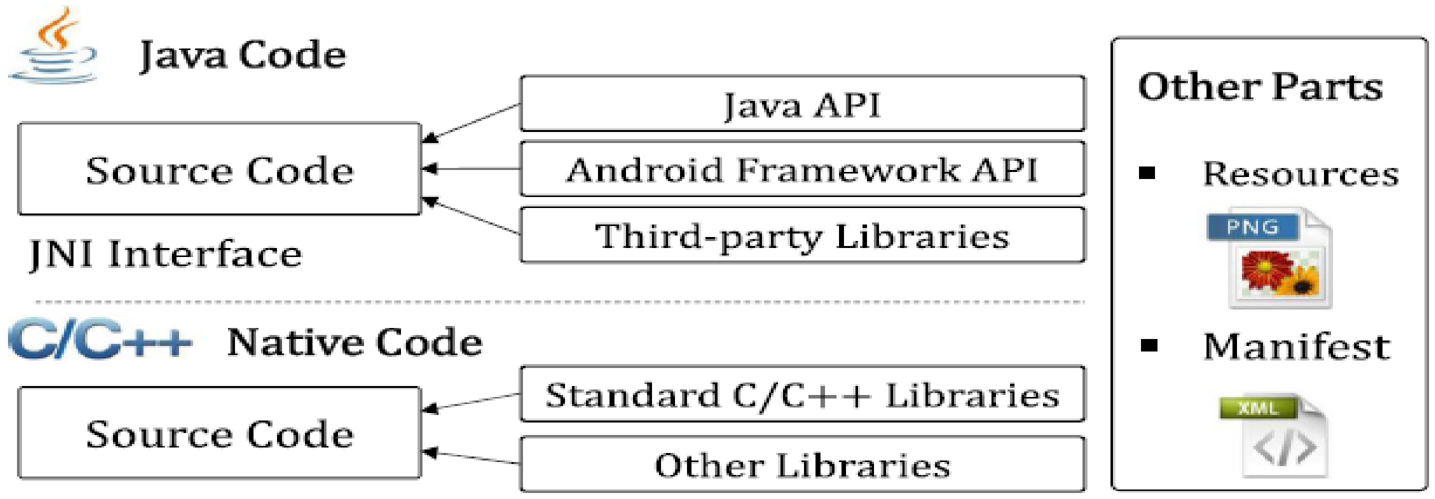
28 March 2025

Outline

- Obfuscation Intro
- Remote MATE Attacks
- Dynamic Security
- White Box Cryptography

Obfuscating Mobile Apps

- Android apps contain different components:
  - Java codes, native codes, third-party libraries, and other resources
  - Applying some obfuscations in an ad-hoc way can achieve limited obscurity
  - The remaining un-obfuscated information could jeopardize the obfuscated code



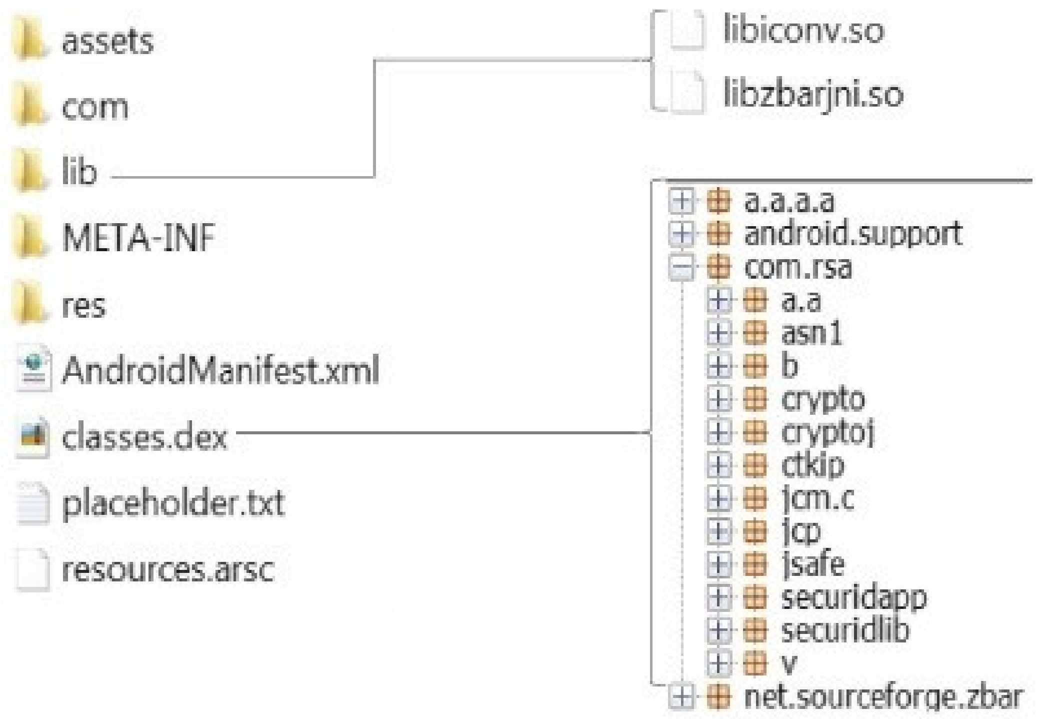
Android Obfuscators

Ca' Foscari University of Venice

- Real-world obfuscation practice usually adopts one obfuscation technique or combines several techniques in an ad-hoc way
- ProGuard is the most popular obfuscation tool for Android apps
  - It is the default one embedded in Android Studio for free use
- ProGuard can only transforms identifiers to meaningless strings
  - Attackers can recover a significant portion of the original lexical information leveraging on the residual information in the obfuscated app (Bichsel et al.)
- DexGuard and DexProtector are more powerful
  - only less than 0.16% of real-world apps employ premium obfuscation tools

Android App Structure Example

- The components of an Android app installation package (e.g. RSA SecureID Software Token)
- All Java classes coded by developers are in the classes.dex component
  - It implements user interfaces (UI) based on the APIs of Android framework and JDK
  - Other UI-related materials (layout, images, and texts) are in the res folder
- The app also employs native codes in the lib folder
  - better than Java bytecodes for implementing some features (security, graphics, performance-critical code)
- There is a manifest file and other folders to store data, such as licenses and fonts.

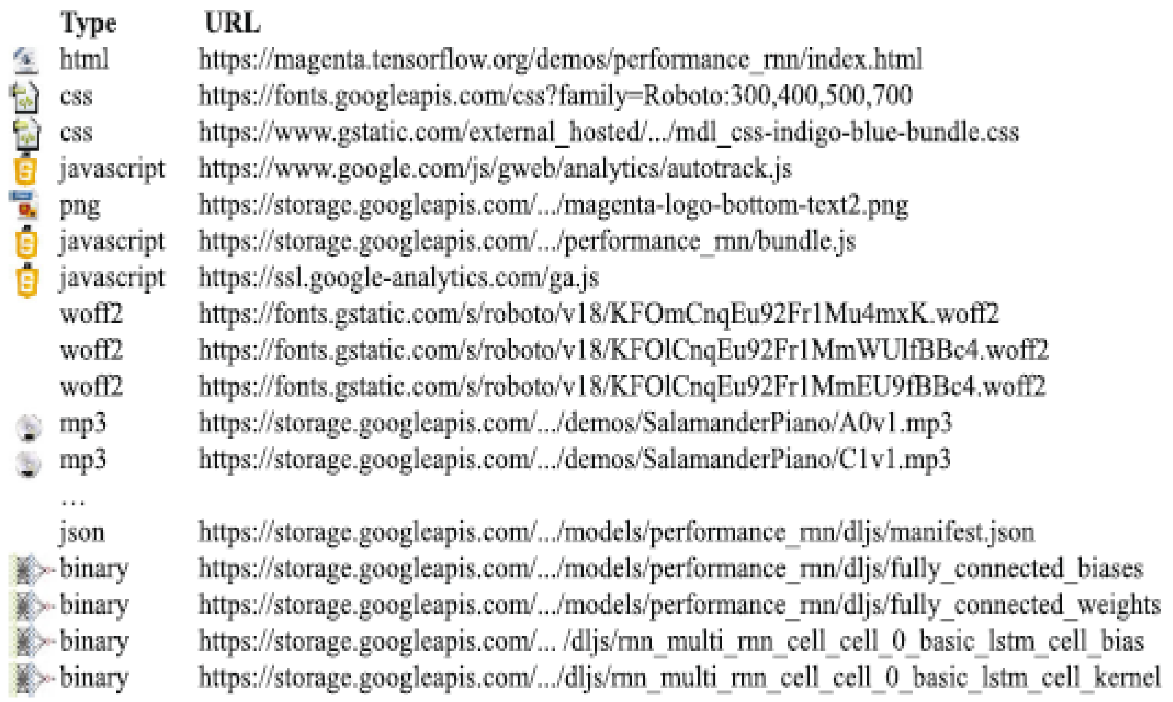


Android App Example

- Because the main feature of the app is to generate one-time passwords, the corresponding password generation codes and seeds should be most critical for protection.
- Current mainstream obfuscation techniques (e.g., lexical obfuscation and control-flow obfuscation) mainly focuses on Java or native codes
- DexProtector can also:
  - use Method Proxy to obfuscate native system calls (libc libraries)
  - encrypt the resource files of software packages and implement functions to decrypt them during runtime.

JavaScript Application Example

- Components of a Javascript application
  - Performance RNN – http://js.tensorflow.org
- Primary index.html file
- CSS file and pics defining appearance
- Bundle.js implements deep learning algorithm
- Binary files define RNN model
- Mp3 files for each note of a piano



Type	URL
html	https://magenta.tensorflow.org/demos/performance_rnn/index.html
css	https://fonts.googleapis.com/css?family=Roboto:300,400,500,700
css	https://www.gstatic.com/external_hosted/.../mdl_css-indigo-blue-bundle.css
javascript	https://www.google.com/js/gweb/analytics/autotrack.js
png	https://storage.googleapis.com/.../magenta-logo-bottom-text2.png
javascript	https://storage.googleapis.com/.../performance_rnn/bundle.js
javascript	https://ssl.google-analytics.com/ga.js
woff2	https://fonts.gstatic.com/s/roboto/v18/KFOmCnqEu92Fr1Mu4mxK.woff2
woff2	https://fonts.gstatic.com/s/roboto/v18/KFOlCnqEu92Fr1MmWUlfBBc4.woff2
woff2	https://fonts.gstatic.com/s/roboto/v18/KFOlCnqEu92Fr1MmEU9fBBc4.woff2
mp3	https://storage.googleapis.com/.../demos/SalamanderPiano/A0v1.mp3
mp3	https://storage.googleapis.com/.../demos/SalamanderPiano/C1v1.mp3
...	
json	https://storage.googleapis.com/.../models/performance_rnn/dljs/manifest.json
binary	https://storage.googleapis.com/.../models/performance_rnn/dljs/fully_connected_biases
binary	https://storage.googleapis.com/.../models/performance_rnn/dljs/fully_connected_weights
binary	https://storage.googleapis.com/... /dljs/rnn_multi_rnn_cell_cell_0_basic_lstm_cell_bias
binary	https://storage.googleapis.com/.../dljs/rnn_multi_rnn_cell_cell_0_basic_lstm_cell_kernel

Obfuscating JavaScript

- Key assets of the program should be in the RNN model and related algorithms
- Thus the binary files and the bundle.js must be obfuscated
- It may further randomize the names of the mp3 files to confuse reverse engineers
- Some tools for javascript:
  - http://stunnix.com/prod/jo/
  - http://jscrambler.com
  - https://www.javascriptobfuscator.com/
  - https://www.obfuscator.io/
  - https://javascript-obfuscator.org/

Obfuscating Neural Networks

- The structure of neural networks is a critical factor to improve the accuracy of deep learning models.
- The structural information of private machine learning models is a key intellectual property for such software
- To obfuscate deep learning models, Xu et al. proposed a simulation-based obfuscation method:
    - The method distils the knowledge of well-trained deep learning models and reloads such knowledge into less deep, shallow networks.
    - In this way, the shallow networks retain the same accuracy as the original models, but they have poor learning abilities.
    - Attackers can learn very few useful settings from the simulation networks

Obfuscation in DRM Systems

- A DRM system controls the access of users to multimedia files.
- The favourite solutions of DRM systems are based on content encryption.
- The critical challenge is to hide decryption keys, especially when attackers can have full access to the decryption software and the computing environment.
- *White-box encryption* is an obfuscation approach which can resist to key extraction attacks (Chow et al)

Other Protection Tools

- DexGuard
- DexProtector
- Themida
- Rewolf-x86-Virtualizer (open source)
- VMProtect
- Code Virtualizer
- Zelix

Diablo

- Diablo is a retargetable link-time binary rewriting framework.
- It can be used for compiler optimization
- It can be used for binary code protections on various platforms

https://diablo.elis.ugent.be/

Performance

Metric	Program	Slowdown
absolute time	application	<1s
relative	application	1.5x
relative	security kernel	100x-1000x

Code virtualizer	ExeCryptor	VMProtect	Themida
100x	700x	500x	1200x

Liem, Gu, Johnson: A compiler-based infrastructure for software-protection, PLAS'08

Time-to-Crack Matters

Ca' Foscari University of Venice

Program	Adversary	Time
hw+sw		many years
well protected	highly skilled, motivated	4-6 weeks
VMProtect	experienced reverse engineer	12 month
mass market malware		minutes-hours

Time-Limited Protection

- Obfuscation provides time-limited protection
- An adversary will require greater-than-zero length of time to extract an asset from an obfuscated program.
- How can we get useful levels of protection from individual transformations that only provide time-limited protection?

Hohl, Time Limited Blackbox Security: Protecting Mobile Agents From Malicious Hosts

References

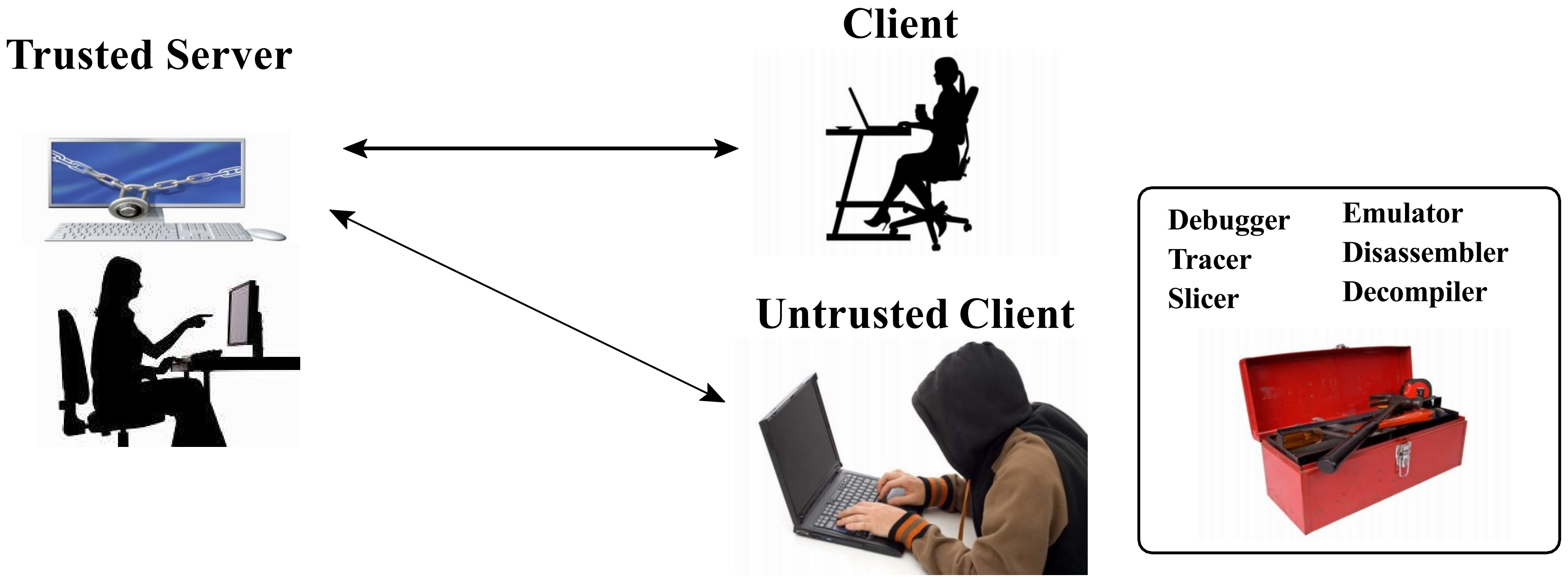
- Bichsel B, Raychev V, Tsankov P, Vechev M (2016) Statistical deobfuscation of android applications. In: CCS 2016. https://doi.org/10.1145/2976749.2978422
- Wermke D, Huaman N, Acar Y, Reaves B, Traynor P, Fahl S (2018) A large scale investigation of obfuscation use in google play. arXiv preprint arXiv:1801.02742. https://doi.org/10.1145/3274694.3274726
- Xu H, Su Y, Zhao Z, Zhou Y, Lyu MR, King I (2018) Deepobfuscation: Securing the structure of convolutional neural networks via knowledge distillation. arXiv preprint arXiv:1806.10313
- Chow S, Eisen P, Johnson H, Van Oorschot PC (2002) White-box cryptography and an AES implementation. In: International Workshop on Selected Areas in Cryptography. Springer. https://doi.org/10.1007/3-540-36492-7_17
- Chow S, Eisen P, Johnson H, Van Oorschot PC (2002) A white-box DES implementation for DRM applications. In: ACM Workshop on Digital Rights Management. https://doi.org/10.1007/978-3-540-44993-5_1

Remote MATE Attacks

Credits: Prof. Christian Collberg

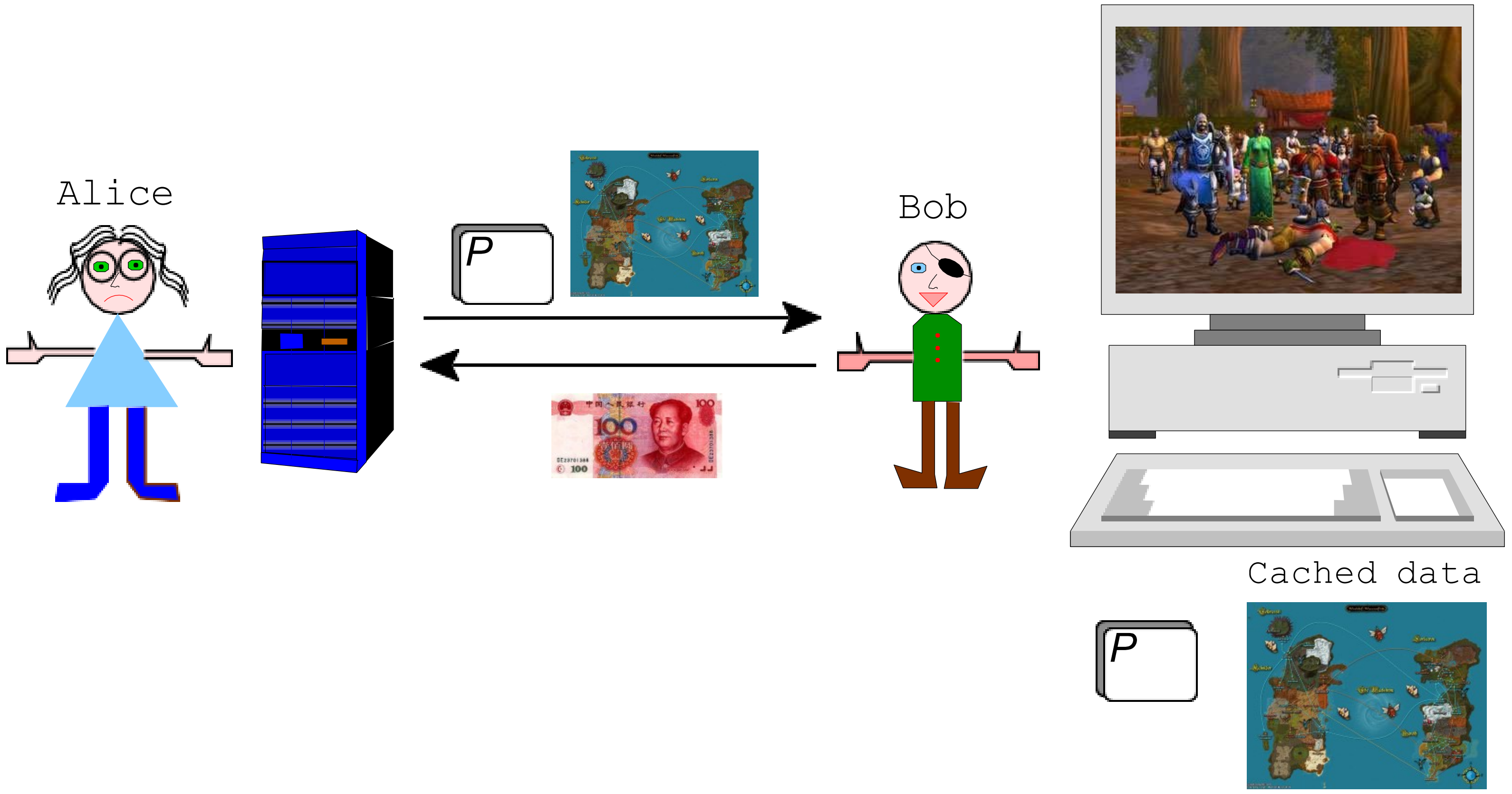
Remote MATE Attacks

- R-MATE attacks occur in distributed systems
  - where untrusted clients are in frequent communication with trusted servers over a network
  - where a malicious user can get an advantage by compromising an untrusted device.



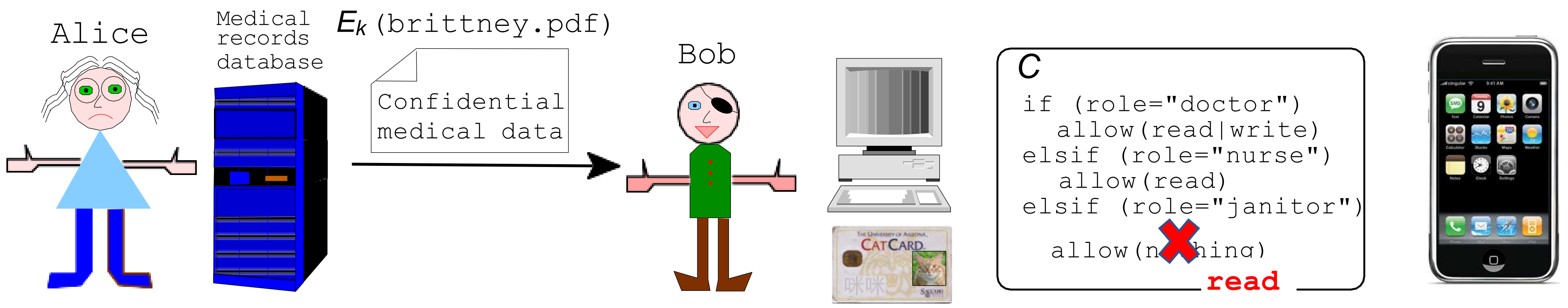
Protecting networked video games

- Prevent Cheating or game cracks from malicious users



Protecting medical records

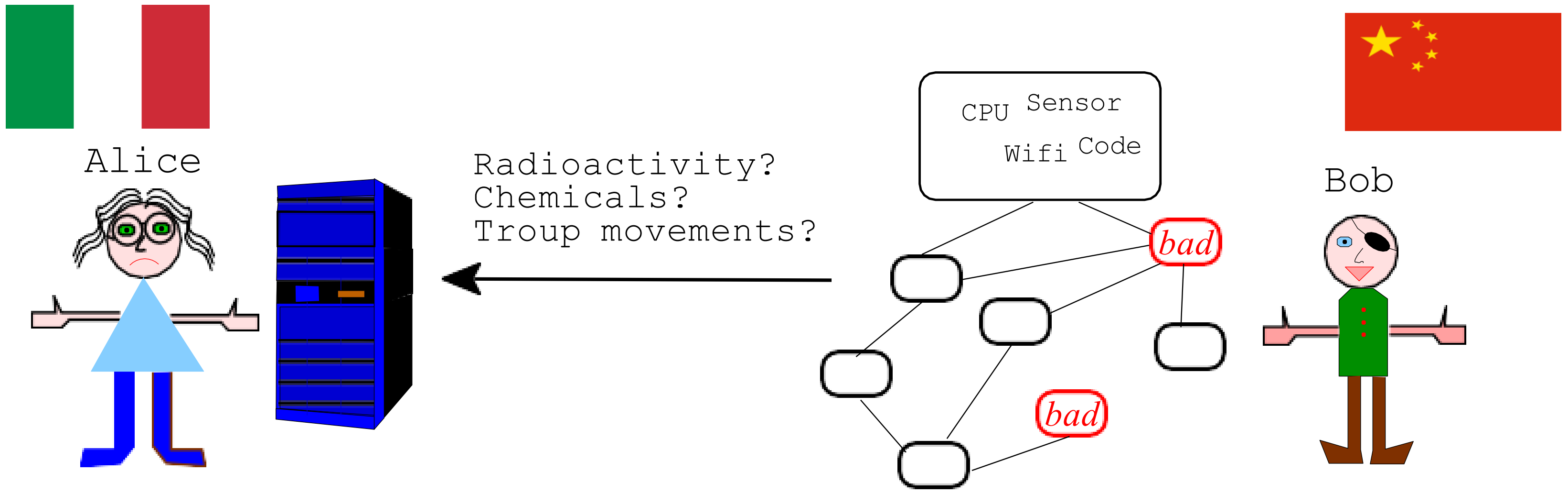
- Medical records must be protected from improper access and improper modification.
- Records are stored on one secure site, accessed from multiple (sometimes mobile) devices.



```
if (role="doctor")
   allow(read|write)
elsif (role="nurse")
   allow(read)
elsif (role="janitor")
   allow(nothing)
```

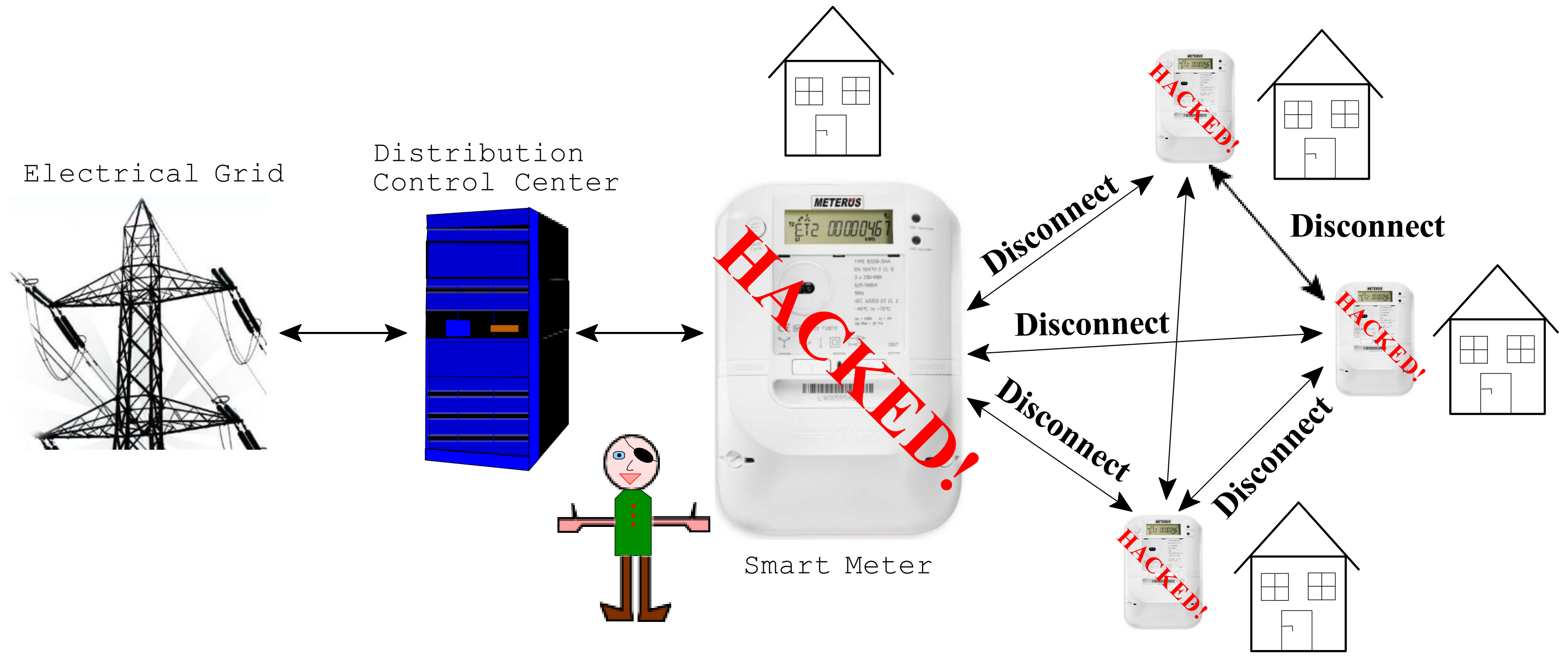
Wireless sensor networks

- Sensor networks are common in military scenarios.
- The enemy can intercept/analyse/modify sensors.



Advanced Metering Infrastructure

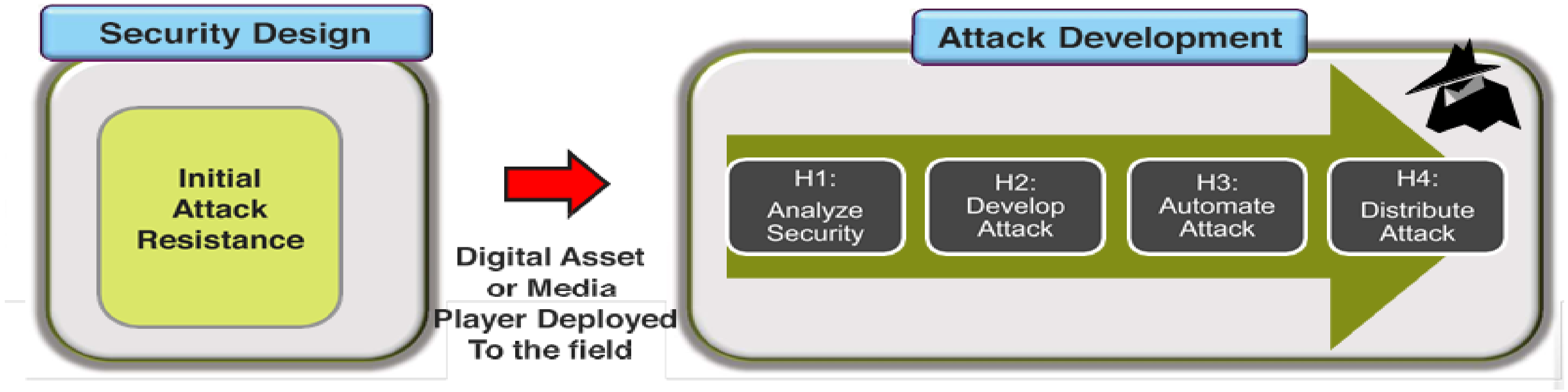
- Selective black-outs, consumers can adjust usage based on current costs, small-scale energy production, . . .



Dynamic Security

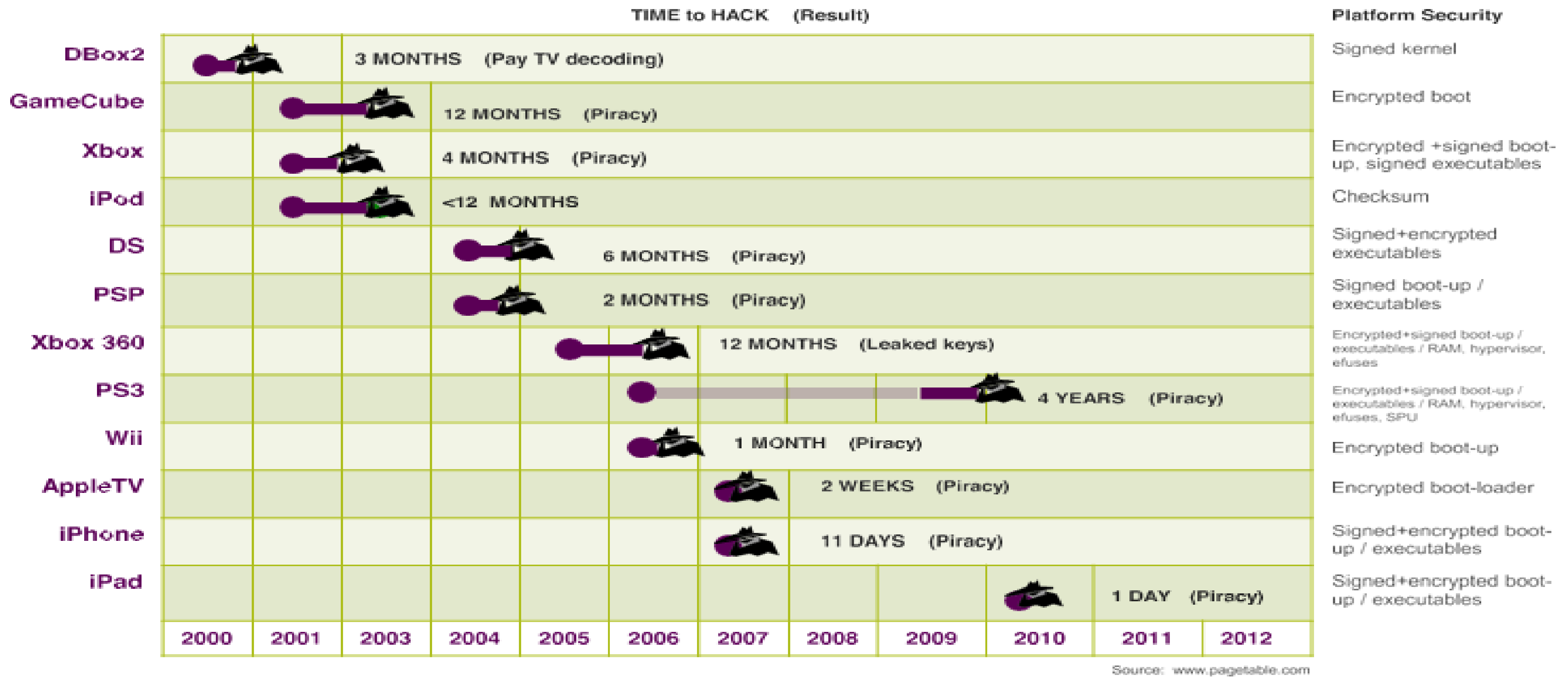
Static Security

- Typical approach to security is to assume that the initial design will remain secure over time
- Anything can be hacked given enough time and effort
  - Set top boxes, C apps, Mobile devices
- Content Owners want to know
  - How will you limit potential damages if there is a breach?
  - What is your renewability strategy?



Static Security

All popular platforms that rely on static security have been permanently hacked

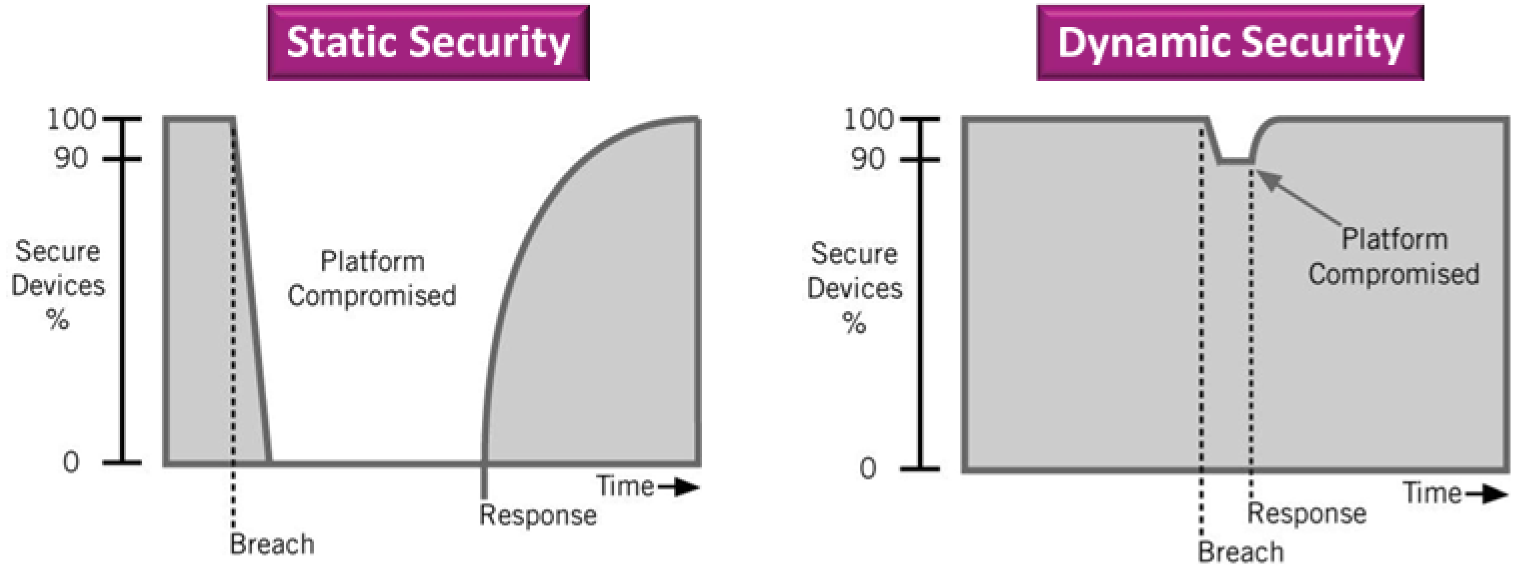


What is Dynamic Security?

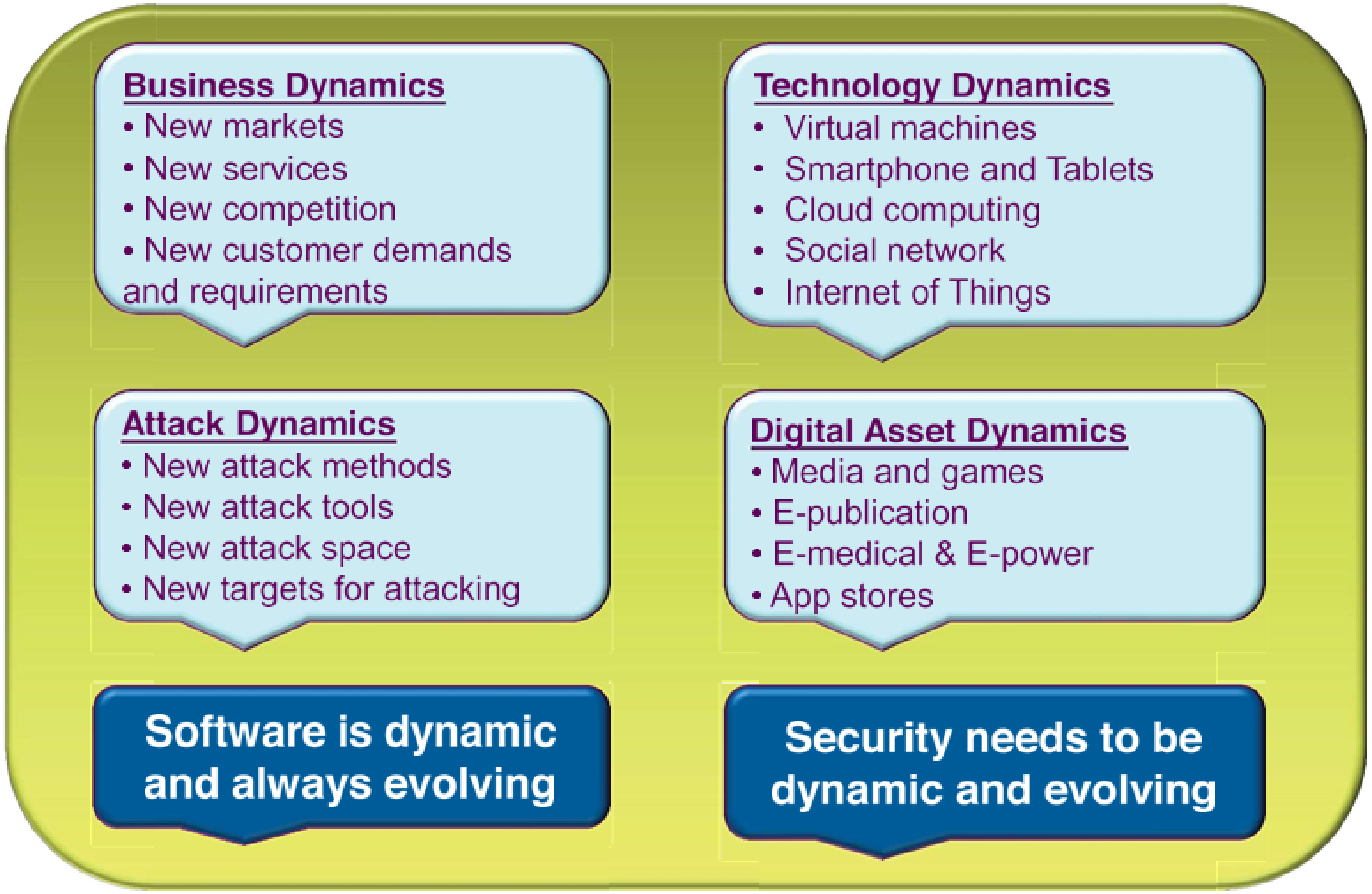
- It is a security model that enables the protection of digital assets against unauthorized use through the upgrade and renewal of the underlying security in the field.
- Proactive Prevention
  - Monitor hacker channels to understand attack techniques
  - Apply security updates to reset hacker’s clock
- Reactive reduction
  - Limits the impact of a breach before it has a big impact
- Benefits
  - Mitigate breach impact and business disruption
  - Thwart potential hacks before they happen

Static vs Dynamic Security

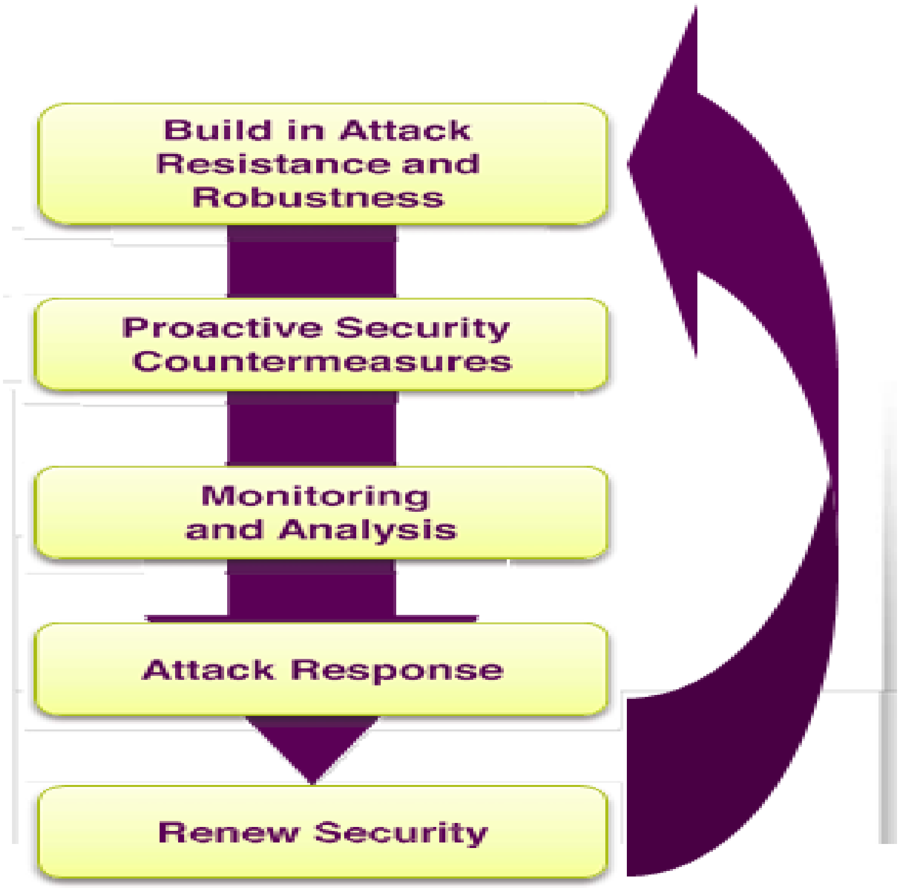
- Once static security breaks, the entire security is gone and hard to be restored
- Once dynamic security breaks, the security can be renewed and restored immediately in a planned way



Dynamic Security is Essential



Security Lifecycle Management


Build in Attack Resistance and Robustness
Proactive Security Countermeasures
Monitoring and Analysis
Attack Response
Renew Security

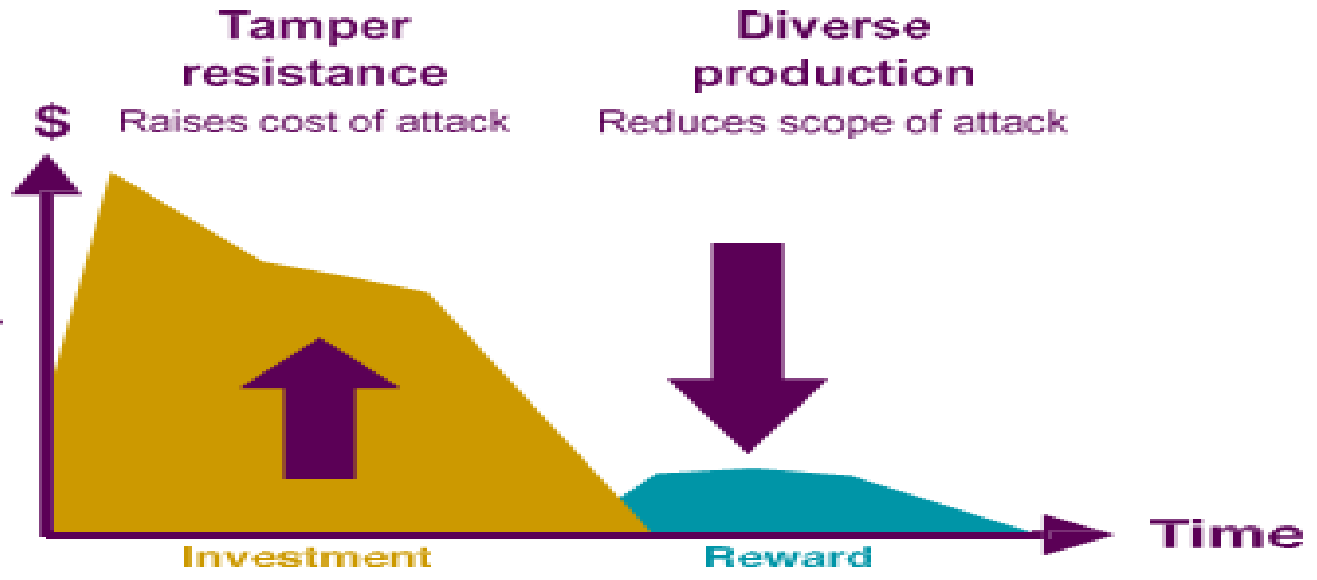
Software Protection

Deploying Obfuscation

- Monitor adversarial communities
- Be prepared with new technologies
- Give adversaries a diversity of targets
  - Spatial diversity
  - Temporal diversity
  - Semantic diversity
- Remote Attestation

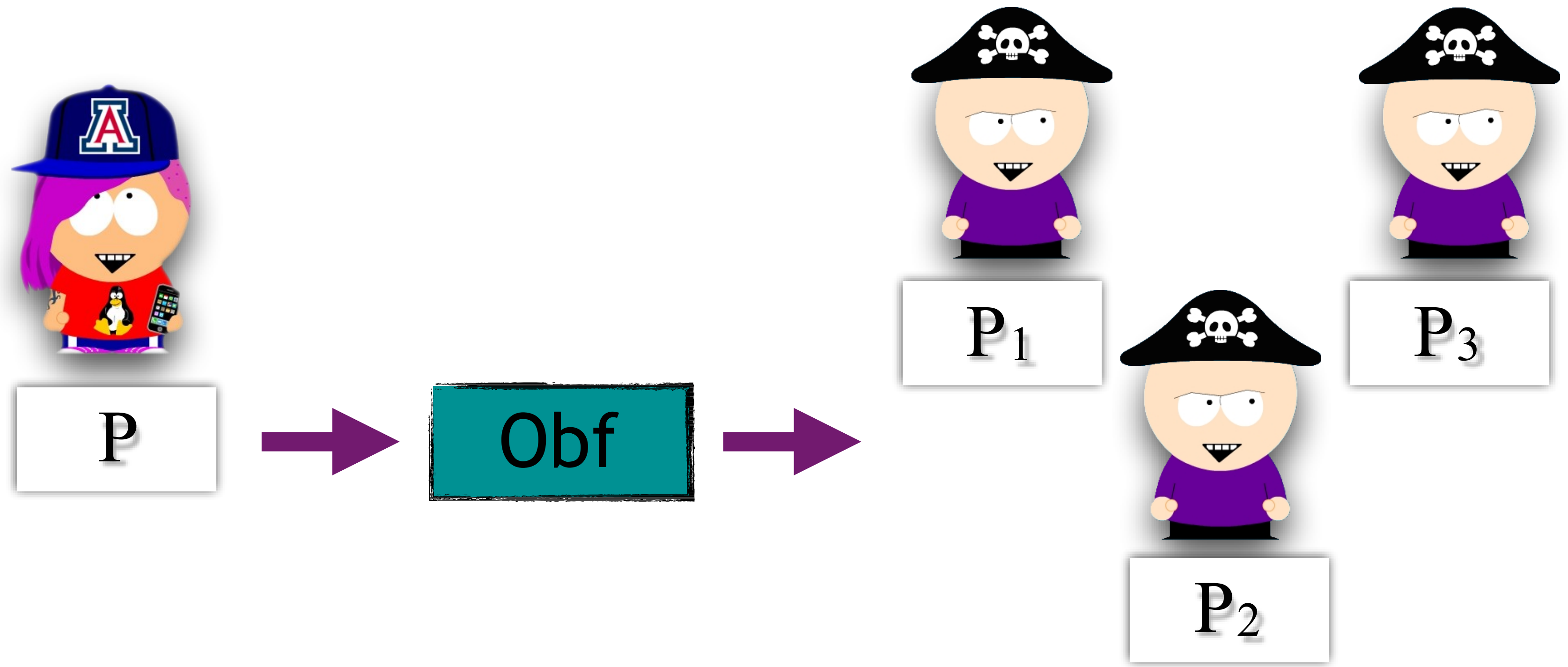
Software Diversity

- Minimize scope of attack
- Prevent automated attacks
- Provide rapid recovery in the event of an attack
- Make the business unattractive to the hacker



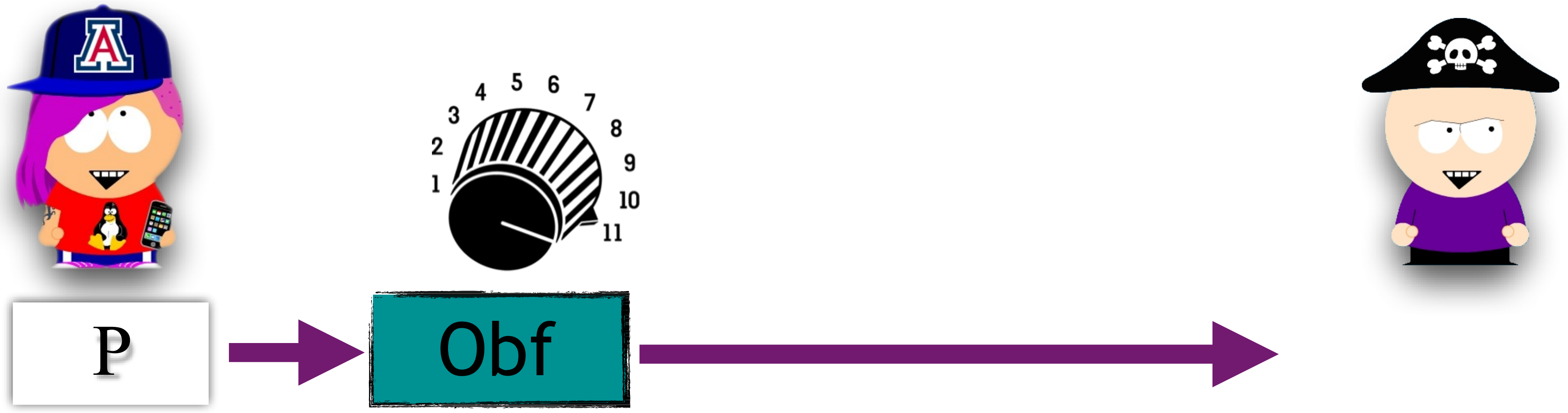
Spatial Diversity

- Prevent collusion by giving each adversary a differently obfuscated program



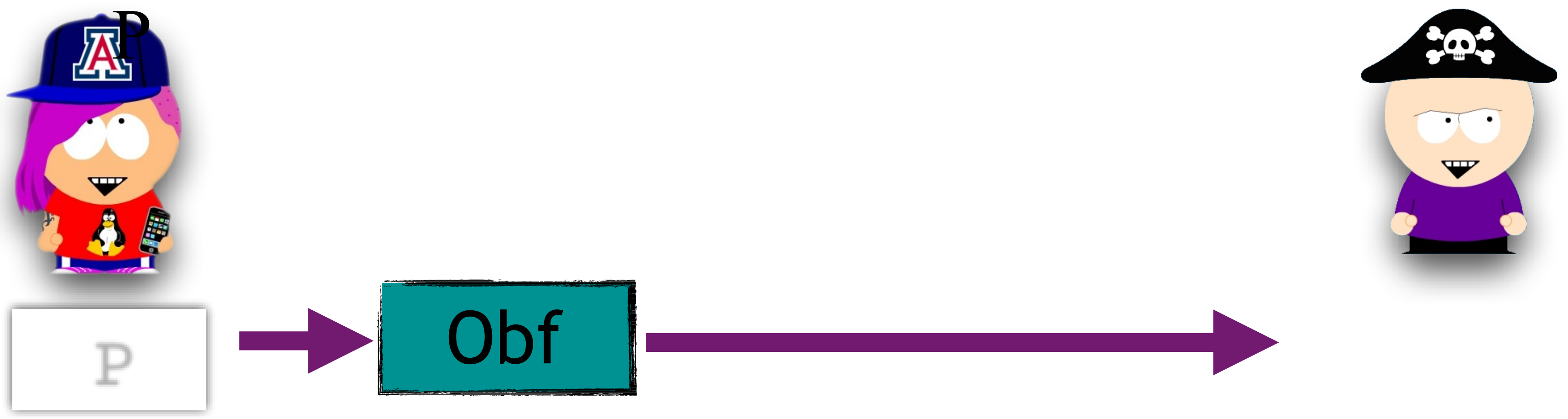
Temporal Diversity

- Adversary sees a sequence of code variants over time
- Overwhelm his analytical abilities
- Small time window to execute an attack



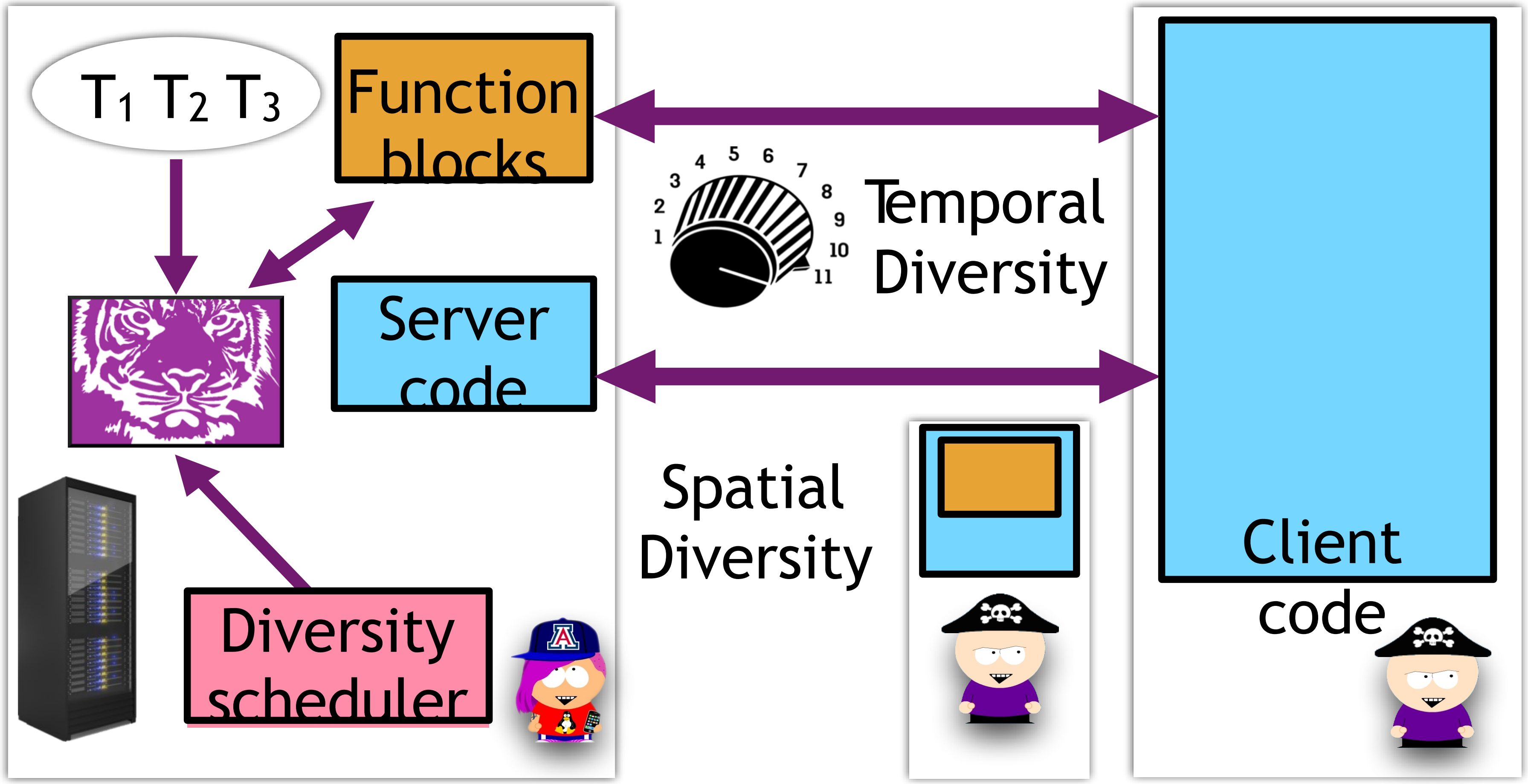
Semantic Diversity

- Code variants are semantically incompatible
- Previously cracked code variants have no value
- Known as “Software Aging”



Jakobsson, et al., Discouraging Software Piracy Using Software Aging, S&P in DRM, 2002

Continuous Replacement



Collberg, et al., Distr. app. tamper detection via continuous softw. updates, ACSAC'12

Remote Attestation

- Extending the idea of Code Guards by sending hashes to a trusted server
  - Decouple hash calculation from hash verification
- "Prove your integrity” challenges
- Challenge = mobile code sent from trusted entity to the client
  - run a different testsuite on the App code and O.S. System and collect results
  - Results checked by the trusted server to detect tampering

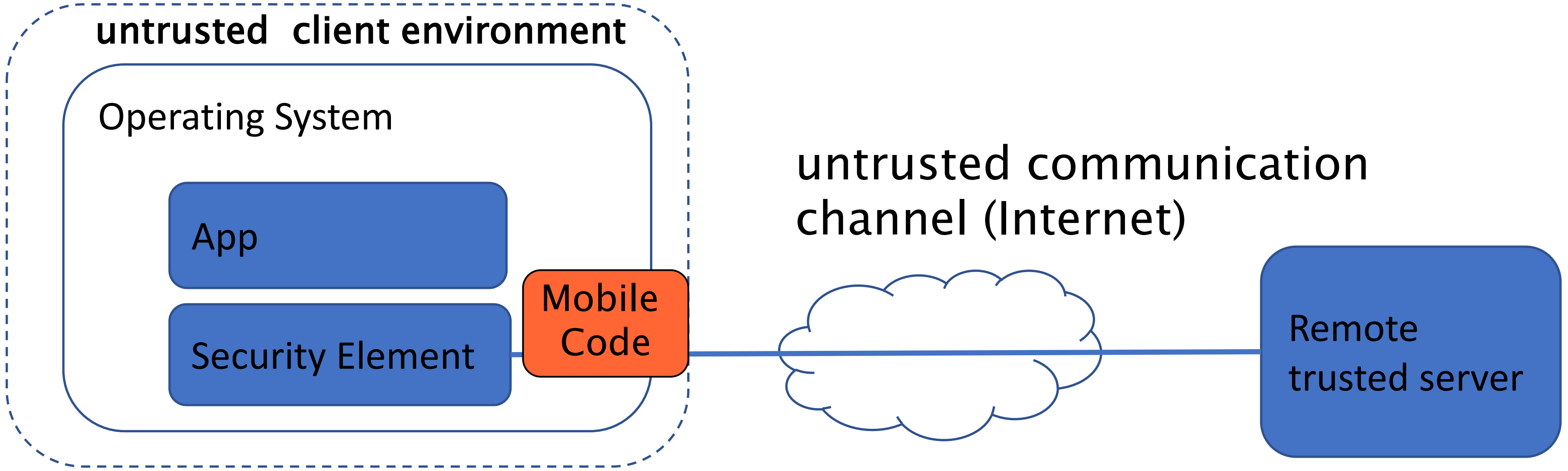
Renewable Remote Attestation

- Tamper-detection code (Self-checkers) are usually embedded in code but they might be removed/ disabled
- Mobile code sent from server linked at run-time
  - to renew self-checkers of binary integrity
- Prototype based on JVM TI extension

P. Falcarin, R. Scandariato, and M. Baldi, "Remote trust with aspect oriented programming," in IEEE Int. Conf. on Advanced Information and Networking Applications (AINA-06), 2006.
R. Scandariato, Y. Ofek, P. Falcarin, and M. Baldi, "Application-Oriented Trust in Distributed Computing," in IEEE Int. Conference on Availability, Reliability and Security (ARES-2008).

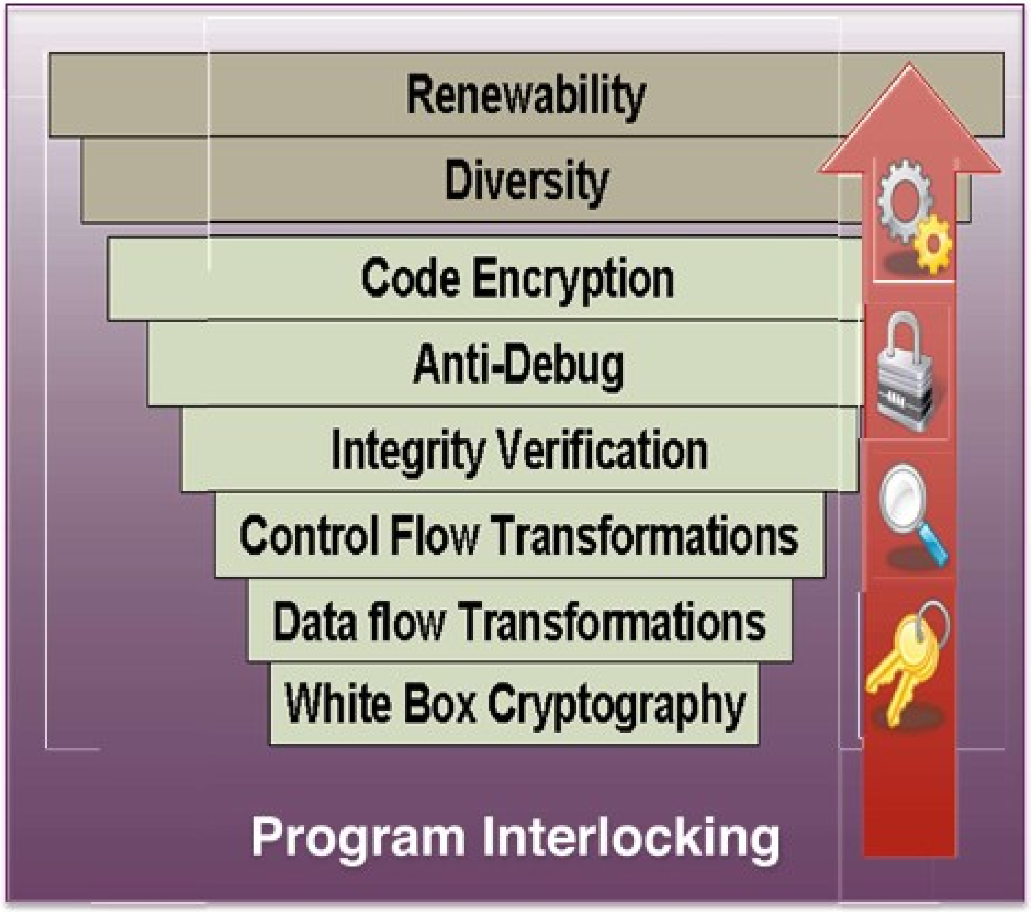
Network-based protections & Mobile Code

- Code mobility makes more difficult to tamper with the code
- Mobile Code from Remote trusted server can be applied on
  - App ➔ Dynamic Binary Obfuscation
  - Op. System ➔ Remote Attestation
  - Security element ➔ Renewable protections (self-checkers)


untrusted client environment
Operating System
App
Security Element
Mobile Code
untrusted communication channel (Internet)
Remote trusted server

Multi-Layer and Interlocked Protection

- Protection application code against collection of attacks
- Provides a multi-layered and interlocked defences
- Flexible and modular to choose the right combination of defences
- Making Security inseparable from your software



Conclusions

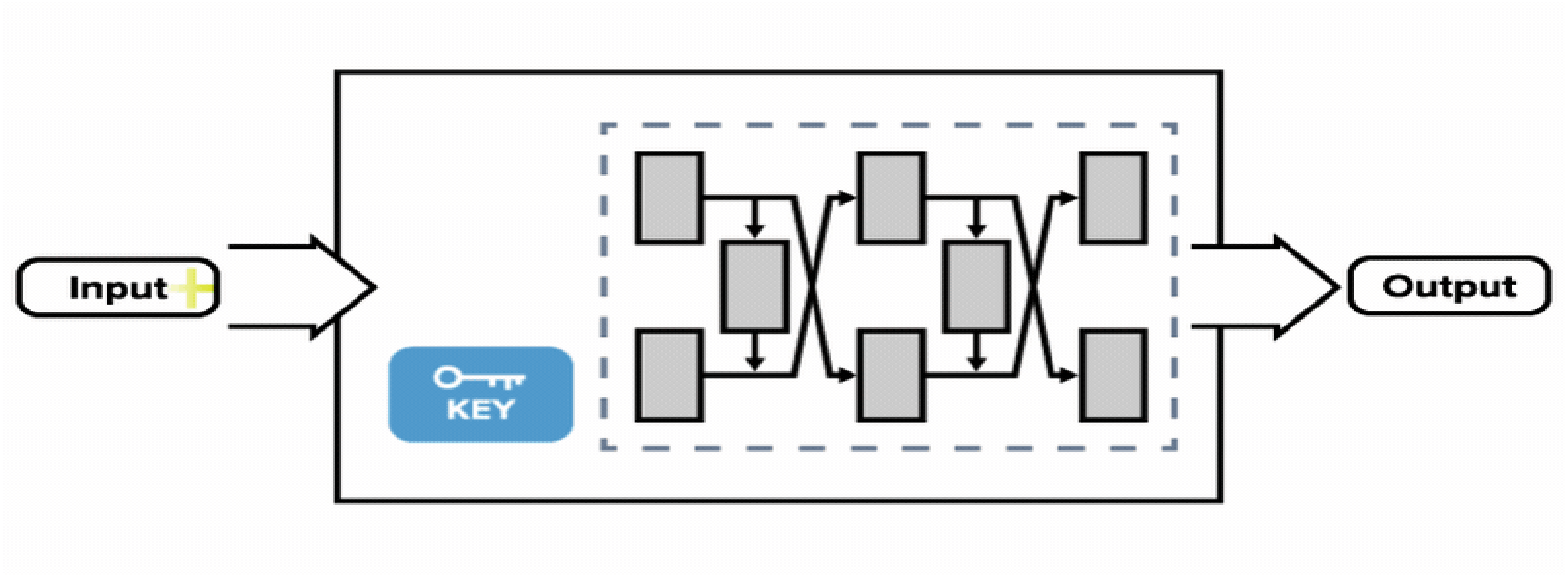
- Scenarios where obfuscation can be useful
- Obfuscating transformations that give time-limited protection
- Updatable security for longer-term protection

- But, how do we know we’re doing anything good?

White Box Cryptography

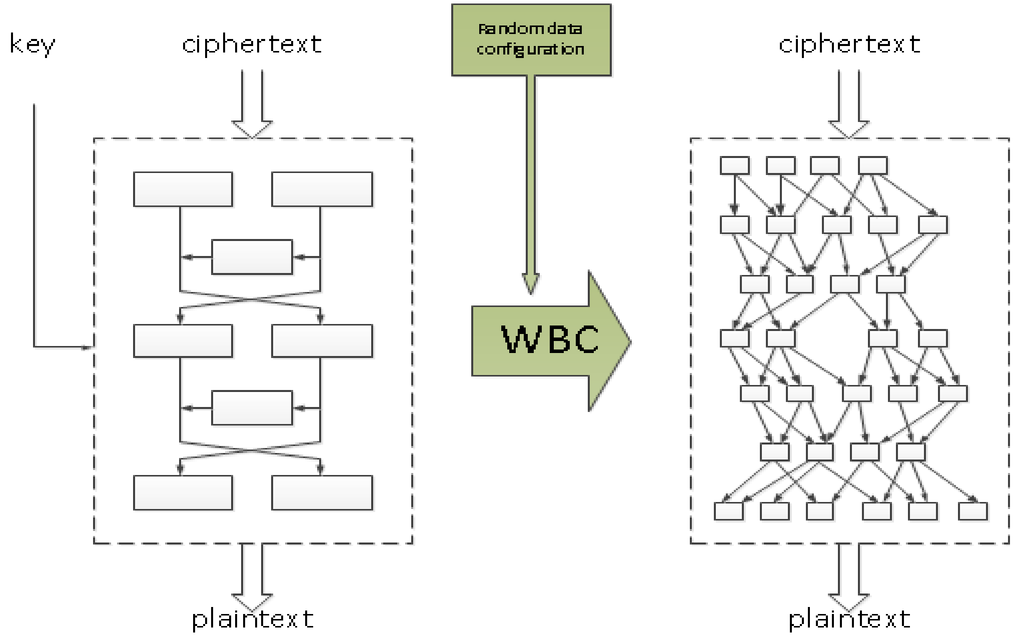
The problem

- A standard encryption algorithm requires that the key be present.
- The key is visible in a debugger at run-time.
- Attackers could easily grab secret keys from the binary implementation, from memory, or intercept information that would lead to disclosure at execution time.



White Box Cryptography

- *White-box encryption* is an obfuscation approach which can resist to key extraction attacks (Chow et al)
- Embed the secret key in “haystack” of code and data
- Strong Obfuscation of crypto code

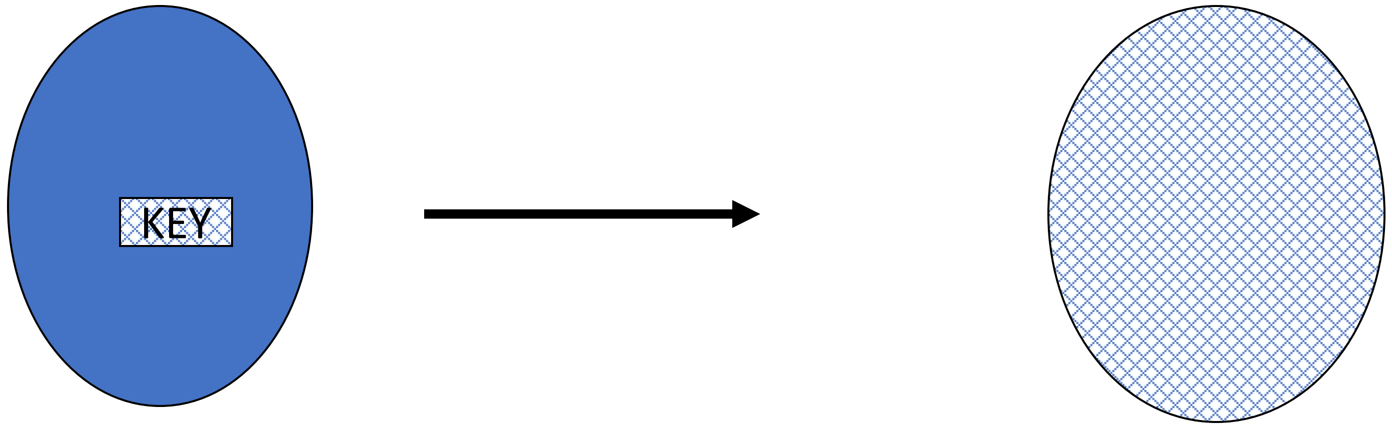


WBC Applications

- Software Agents
  - Embedded cryptographic keys for signing purposes
- Digital Rights Management (DRM)
- Smart Card Technology
- Asymmetric crypto system

White Box Crypto General Idea

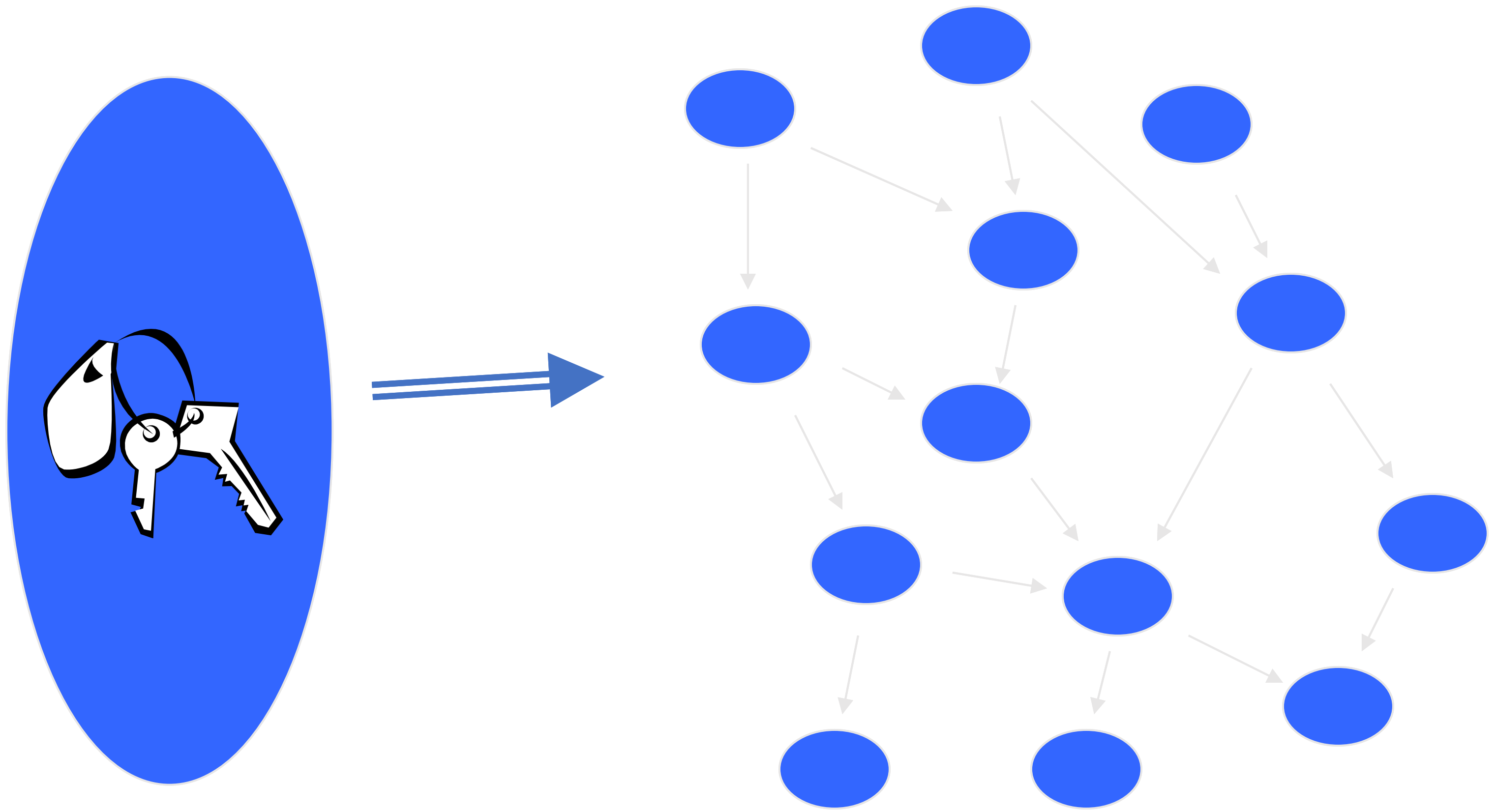
Spreading embedded secret information



Thus, forcing an attacker to understand a greater part of the implementation

How?

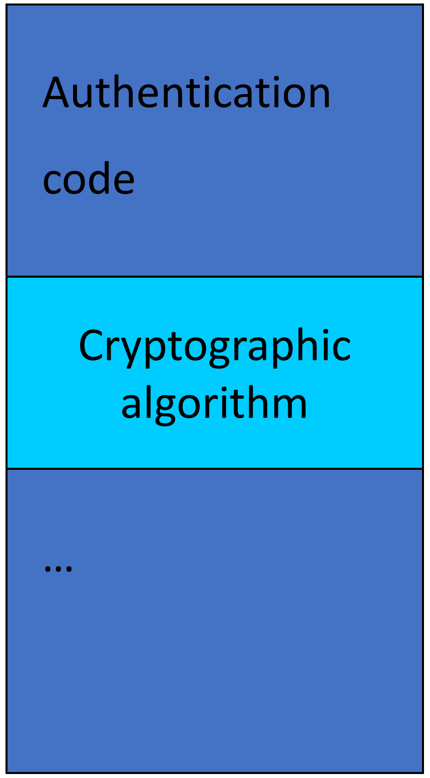
- White-Box
- Transformations



Transform an algorithm into a series of key-dependent lookup tables

WBC General idea

Expanding the cryptographic border



- External function encoding

$$E'_k = g \circ E_k \circ f$$

- Attacker:
  - Analyse E'_k
  - Isolate random bijections g, f
  - Analyse E_k to find k
- Goal: make isolation difficult

White-Box Transformations

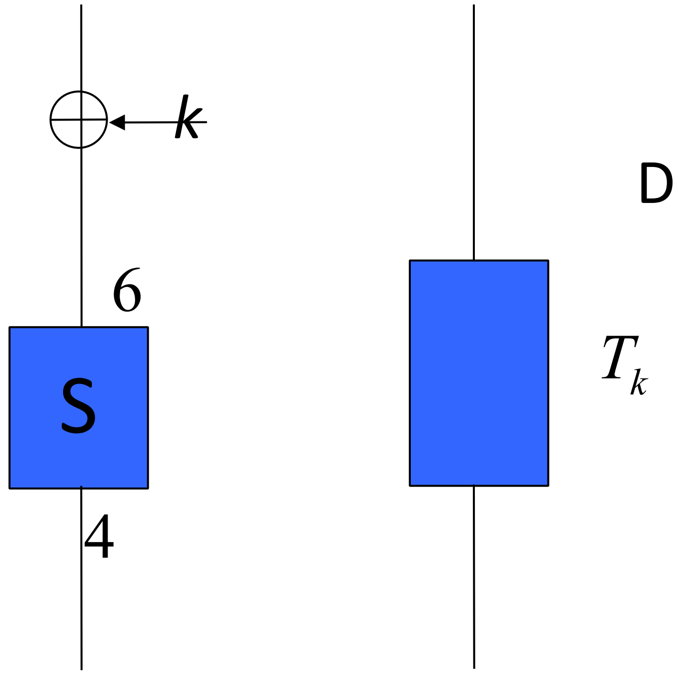
- Partial Evaluation
- Combined Function Encoding
- By-Pass Encoding
- Split Path Encoding
- …

Techniques apply on cryptographic algorithms build with XOR, substitution and permutation functions

- AES, DES, …

White-Box Transformations (2)

- Partial Evaluation

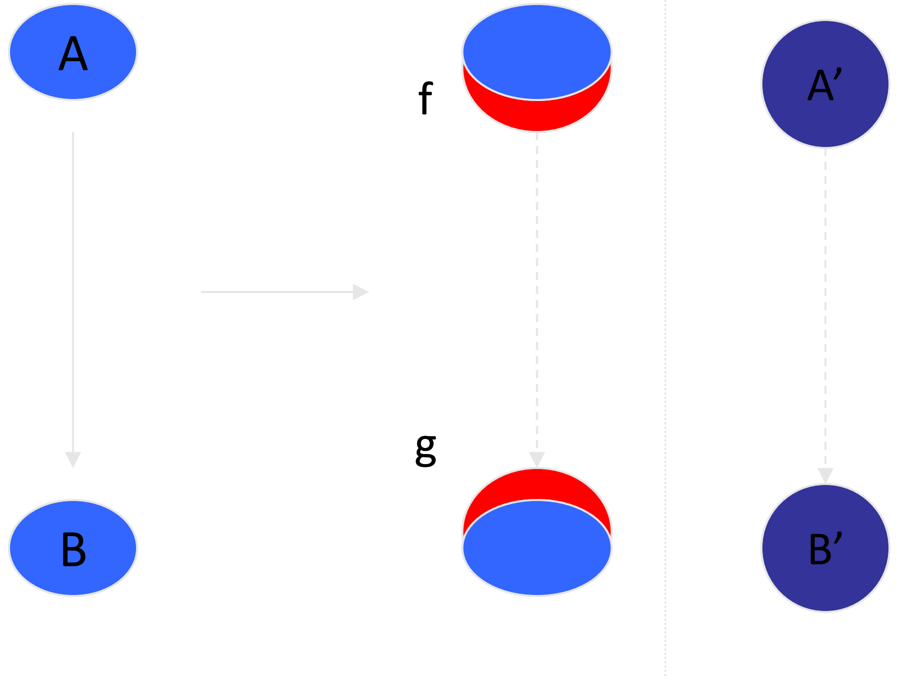


$$T_k = S(x \oplus k)$$

Definition of a new key-dependent lookup table

White-Box Transformations (3)

- Internal Function Encoding



Choose random bijection f

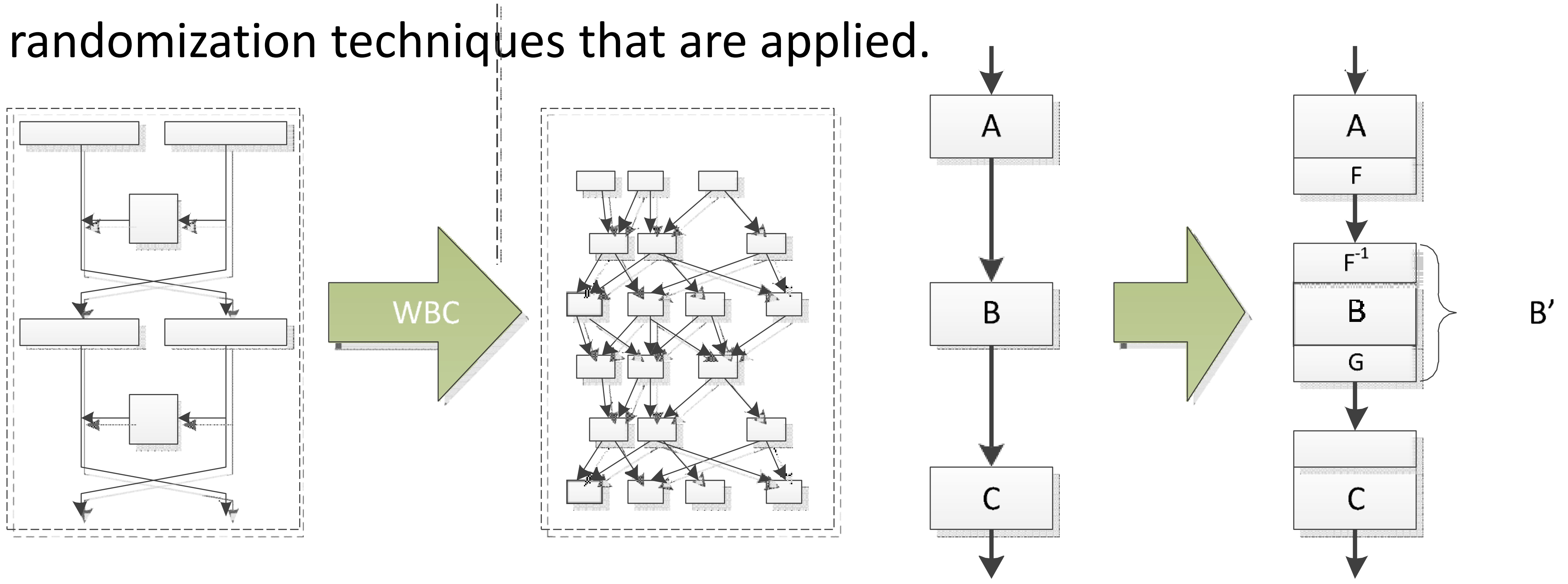
and $g = f^{-1}$

Encoded version:

$$\begin{cases} A' = f \circ A \\ B' = B \circ g \end{cases}$$

Chow et al. WB implementation

- The secret key is hard-coded into the lookup tables and protected by randomization techniques that are applied.



(a) Transformation into network of lookup tables

(b) Encoded lookup tables

White Box Encryption in DRM Systems

- White-box encryption approach pre-evaluates all the operations related to keys and replaces corresponding codes.
- Each function XORs the plaintext with a round key, and then employs a lookup table and a permutation box to produce the output.

Some Numbers

- DES
  - Chow et al.: 4,54 Mb
  - Improvement by Link et al.: 2,25 Mb
- AES
  - Normal implementation: 4.352 bytes
  - Chow et al.: 770.048 bytes

    177 times bigger, 55 times slower

    3104 lookups

Local Security

- Internal function encoding provides local security
- A’ is known. Because the bijection f is random, no information can be revealed of A
- Similar to one time path

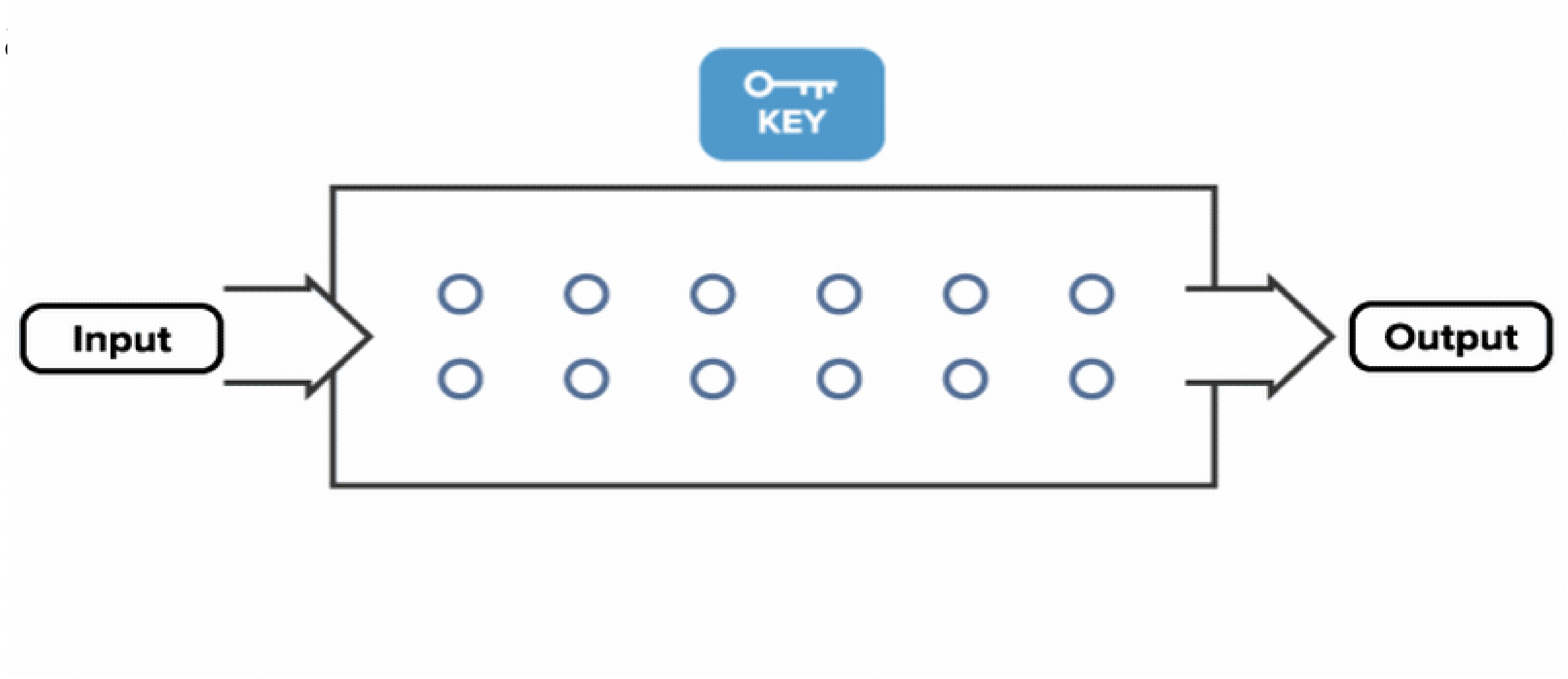
Global Security

- Currently no proof
- Can we guarantee white-box security?
- Trade-off between performance and level of security

- AES: Cryptanalysis by Billet et al. (2004)

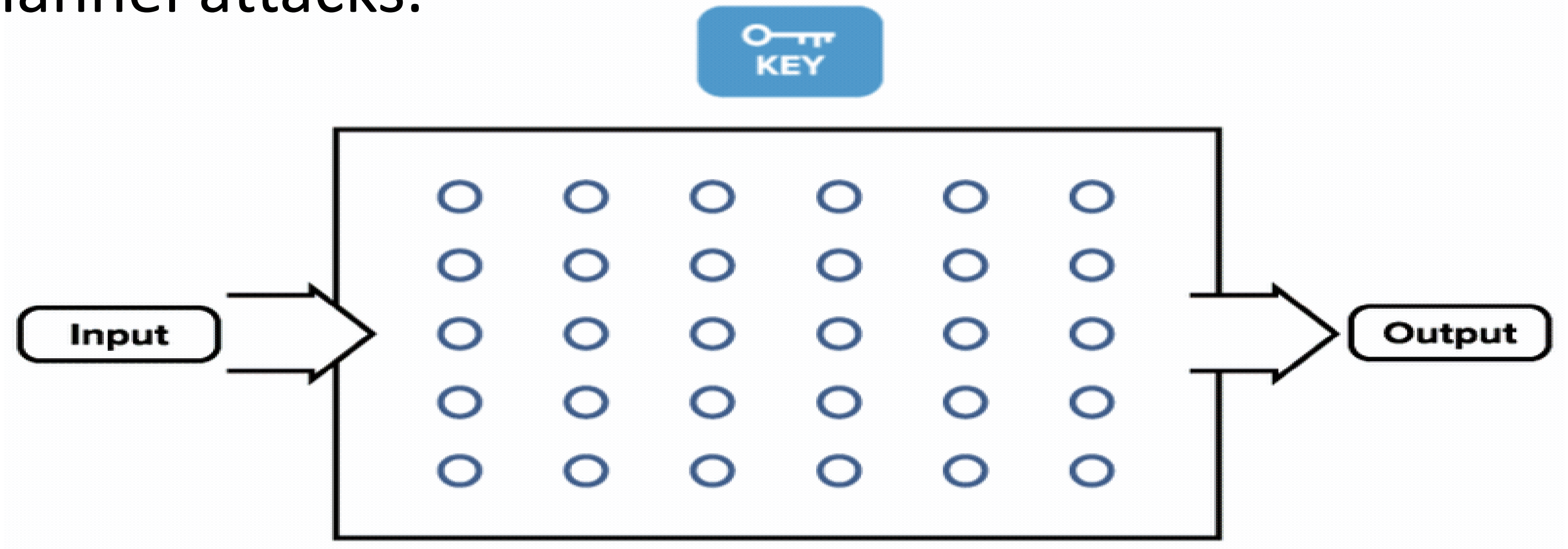
White-Box Cryptography recap

- White-box technology hides the key.
- 2nd generation white-box tools attempt to randomize execution behaviour (Chow et al., 2002)
- The basic flow of the algorithm is unchanged and is vulnerable to side-channel



White-Box Cryptography

- New 3rd generation white-box tools hide the key and randomise **ALL** execution behaviours.
- They provide greater depth and strength of security and are not vulnerable to side-channel attacks.



1. https://paceap.com
2. http://Irdeto.com

WBC - References

- [Chow02DES] Stanley Chow, Philip A. Eisen, Harold Johnson, and Paul C. van Oorschot. A white-box DES implementation for DRM applications. In Proceedings of the ACM Workshop on Security and Privacy in Digital Rights Management (DRM 2002), volume 2696 of Lecture Notes in Computer Science, pages 1–15. Springer, 2002.
- [Chow02AES] Stanley Chow, Philip A. Eisen, Harold Johnson, and Paul C. van Oorschot. White-Box Cryptography and an AES Implementation. In Proceedings of the 9th International Workshop on Selected Areas in Cryptography (SAC 2002), volume 2595 of Lecture Notes in Computer Science, pages 250–270. Springer, 2002
- [Kerins06] Tim Kerins and Klaus Kursawe. A cautionary note on weak implementations of block ciphers. In 1st Benelux Workshop on Information and System Security (WISSec 2006), page 12, Antwerp, BE, 2006.