RSAConference2019
San Francisco | March 4–8 | Moscone Center

SESSION ID: BAC-F01

Key Management & Protection: Evaluation of Hardware, Tokens, TEEs and MPC

Prof. Nigel Paul Smart

KU Leuven
@SmartCryptology

The Problem

- Cryptography is a central tool of computer security
- All cryptography relies on secrets:
  - Key exchange
  - Encryption
  - Digital signing
  - Authentication: passwords, keys, biometric templates
- Once a secret is stolen, all security is lost
  - Anything encrypted can be decrypted
  - All signing protections are lost (e.g., code signing)
  - Users can be impersonated
  - **All of the above can be done without being detected**

The Solution

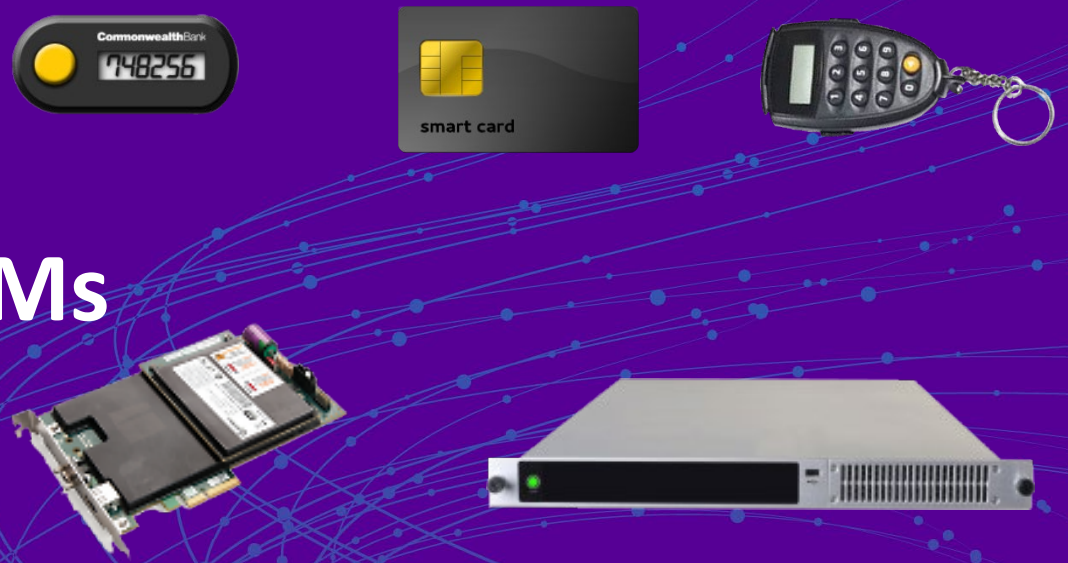
- **Key protection** - prevent an attacker from stealing secrets and keys

- The cat-and-mouse problem – how do we protect keys without having the mechanism that protects them being stolen?

- Two main directions
  - Hardware
  - Software

RSAConference2019

Hardware -
Tokens, Smartcards and HSMs



Hardware – Smartcards and HSMs

- Dedicated hardware for cryptographic operations
- Strong protection of private keys
  - Keys are never exported (only key use is granted)
  - Strong physical protection to prevent key access
  - Special purpose device – no other code can run
  - Certified to ensure compliance

Proc and Cons of Secure Hardware

Pros:

- Clear security guarantee - assuming secure construction
  - Cannot be cloned
  - Keys cannot be stolen

Cons:

- Side Channels (old problem)
- Agility (old problem, becoming more relevant)
- Responding to vulnerabilities
- Usage of high-value keys

Problems with Hardware – Side Channels

- Side-channel attacks (same old attacks, sometimes in new guises)
  - Timing attacks
  - Power analysis and differential power analysis
  - Acoustic analysis
  - Fault attacks

- Early smartcards/tokens suffered significantly from these, but we still see problems in the field
  - Problem still exists (especially for “low end” devices)
  - At BlackHat Asia 2017, presented a break of South Korea Transit Card using Side-channel analysis

- Situation is much better for **high-end** smartcards and for HSMs

Problems with Hardware – Agility

- **Agility – ability to quickly update**
  - Crypto agility – new crypto vulnerabilities (e.g. SHA-1), standards, PQC, etc.
  - Software agility – updates due to software vulnerabilities
- **Degradation to lowest common denominator**
  - HSM A only supports 0-IV CBC mode with 3DES , whereas HSM B supports AES GSM
- **Lack of agility**
  - Padding-oracle attacks were hard to fix
  - New methods are slow to be adopted
    - E.g., cannot run ECDSA signing for Bitcoin inside an HSM since the curve is not supported!

A Note on Vulnerabilities

- Hardware modules are often just software in a closed box
- **All** software suffers from vulnerabilities
  - Bugs in vendor code
  - Bugs in libraries used
  - Bugs in OS
- This is not going to change
  - Software development processes have improved, but systems are increasingly complex and attackers more sophisticated
- In this environment, a factor to consider is how hard it is to recover from the discovery of vulnerabilities
  - Especially when the software is in a closed box you cannot open!

Problems with Hardware – Key Usage

- Key usage is still a big problem
- In some applications, even a single fraudulent key use is a problem
  - Code signing
  - Cryptocurrency
  - Authentication to a sensitive account
- If an attack breaches a machine that can use the keys on the HSM, then this can still be a problem
  - Can one integrate the authorization engine with the signing engine?

IBM

IBM PSIRT Blog

IBM Security Notice Regarding IBM Code Signing Certificates

Oct 14, 2016 7:34 am EST

Share this post:

Two IBM Code Signing Certificates Are Being Revoked

IBM recently has identified malware which appears to have been signed by one of two IBM code signing certificates, one for signing Java code (serial number 20 29 13 7c b7 f0 b6 4b e9 bf 21 1a bc 6e ed 10, which already is expired) and the other for signing code to run on Windows systems (serial number 50 02 c1 5f e2 a1 c1 2e bf 2b 04 35 13 54 ae f6). IBM is revoking these certificates today, October 14, 2016. Updates for products which may have been using these certificates will be published as soon as they are available. To IBM's knowledge, this malware has not been distributed with any IBM software.

Partners Using Carbon Black: Stolen Signing Certificate Leads to Banking Trojan

KIM ZETTER SECURITY 09.27.12 05:56 PM

HACKERS BREACHED ADOBE SERVER IN ORDER TO SIGN THEIR MALWARE

Bitcoin Stolen: How NiceHash Was Robbed Of $78 Million and What's Next

A highly orchestrated heist hits a popular bitcoin mining service.

By Danny Paez on December 8, 2017 Filed Under Hackers & Security

Kobal goes on to explain that in a window of three to four hours the attackers were able to use a NiceHash engineer's credentials to access their network using a VPN. From there they were able to "simulate the workings of [NiceHash's] payment systems," which allowed them to siphon thousands of bitcoins.

RSAConference2019

Securing Keys with General Purpose "Secure" Hardware

Securing Keys with General Purpose "Secure" Hardware

A new emerging technology – trusted execution

- Intel SGX
- ARM TrustZone

Supposed to provide a secure enclave where general purpose code can be run

- Provided with memory encryption
- Code running in enclave cannot be viewed
- Can be used for protection of keys and general data
- Attestation methods to make sure correct code is running

Dedicated Crypto HW vs. General Purpose HW

The isolation issue

- Dedicated HW (HSM) runs nothing but the crypto code, and so no software side channels are possible
- General purpose HW is vulnerable to cache attacks, speculative execution attacks, clock attacks,…

Security without isolation – it is very hard (if not impossible) to prevent side channels

- We don’t know all of the channels yet (cannot prevent what you don’t know)
- Experience has shown that writing secret-independent code is extremely hard, if not impossible
- This makes general purpose HW a cat-and-mouse situation

Software Side Channel Attacks

- Cache attacks
- Speculative execution attacks
- Other side channels
  - Fault attacks via clock speed changes
  - Acoustics
  - It’s an ever-changing area (cat-and-mouse zone)

Lea

Inferring Fine-grained Control Flow Inside SGX Enclaves with Branch Shadowing

[1]SKL Sangho Lee[†] Ming-Wei Shih[†] Prasun Gera[†] Taesoo Kim[†] Hyesoon Kim[†] Marcus Peinado[*]

[†] *Georgia Institute of Technology*
[*] *Microsoft Research*

Abstract

Intel has introduced a hardware-based trusted execution environment, Intel Software Guard Extensions (SGX), that provides a secure, isolated execution environment, or enclave, for a user program without trusting any un-trusted computing bases (TCBs) and a long history of vulnerabilities. Systems like Overshadow, InkTag or Haven attempt to remove the operating system (OS) from the TCB of applications while retaining its functionality. However, the untrusted OS's control of most physical resources puts it in a much better position to launch side-

we need either to fully trust the operator, which is problematic [16], or encrypt all data before uploading them to the cloud and perform computations directly on the encrypted data. The latter can be based on fully homomorphic encryption, which is still slow [42], or on property-

OS can launch deterministic side-channel attacks against protected applications running on SGX. Their attacks can steal documents and outlines of JPEG images from single runs of three legacy applications protected by Haven and InkTag. The attacks are more powerful than traditional side-channel attacks by unprivileged attackers because the untrusted OS retains control of most of the hardware.

Speculative Execution – Meltdown & Spectre

- Modern processors use sophisticated methods to run faster
  - Processor guesses branch and continues by guess
  - If the guess is incorrect, then processor rewinds everything
  - But the cache is not changed

- Attack code can train branch predictor of processor

- Spectre has been shown to work again SGX
  - There are no good mitigations

SGXPECTRE Attacks: Leaking Enclave Secrets via Speculative Execution

Guoxing Chen, Sanchuan Chen, Yuan Xiao, Yinqian Zhang, Zhiqiang Lin, Ten H. Lai
Department of Computer Science and Engineering
The Ohio State University
{*chen.4329, chen.4825, xiao.465*}*@osu.edu*
{*yinqian, zlin, lai*}*@cse.ohio-state.edu*

Abstract

This paper presents SGXPECTRE Attacks that exploit the recently disclosed CPU bugs to subvert the confidentiality of SGX enclaves. Particularly, we show that when branch prediction of the enclave code can be influenced by programs outside the enclave, the control flow of the

clave programs) to harness the SGX protection, while non-sensitive components run outside the enclaves and interact with the system software.

Although SGX is still in its infancy, the promise of shielded execution has encouraged researchers and practitioners to develop various new applications to utilize

RSAConference2019

Software -
Can software provide a solution?

The Appeal of Software-Based Key Protection

- Ease of use and ease of management
  - Both for servers and endpoints
  - Suitable for virtualized environments
  - Users already carry around powerful computers – let's use them for security
    - Mobile phones actually have security advantages over OTP devices and smartcards

- Software is agile by definition
  - Easy to update
    - Bugs and flaws are everywhere, not just hardware
    - Need to have a good update strategy at the onset
  - Easy to incorporate new algorithms, standards, etc.

Securing via Software

- **The ideal – black-box obfuscation**
  - Code that computes the secret operation, but cannot be read (and so key cannot be extracted)

- **Problems**
  - General black-box obfuscation is impossible
    - Barak et al., On the Impossibility of Obfuscating Programs, CRYPTO 2001
    - Weaker versions (iO) may be possible, but completely impractical today
      - So slow it makes general Fully Homomorphic Encryption look practical
  - Although appealing, it doesn't even **solve** the problem
    - If I can copy the code in entirety then I don't even need the key
    - Can be used in theory to enable partial operations (sign on only a type of code)
    - Can try to lock to a specific device (but code can be fooled)

Whitebox Crypto

- General Obfuscation is impossible, and iO looks awful in practice.
- Whitebox Crypto: *Ad-hoc Obfuscation specifically for crypto algorithms*
  - Note: it doesn't solve the problem of copying the entire code (cloning)
- Very appealing – a number of vendors offer Whitebox
  - Excellent for mobile
  - Simple to deploy on endpoint and server
- Initially, was intended for DRM where it makes more sense, but wider applications being found.
- Main question is how secure is it?

Call for participation

The competition comes in two flavors for competitors:

- Developers are invited to post challenge programs that are white-box implementations of AES-128 under freely chosen keys. Challenges are expected to resist key extraction against a white-box attacker.
- Attackers are invited to break the submitted challenges i.e. extract their hard-coded encryption key.

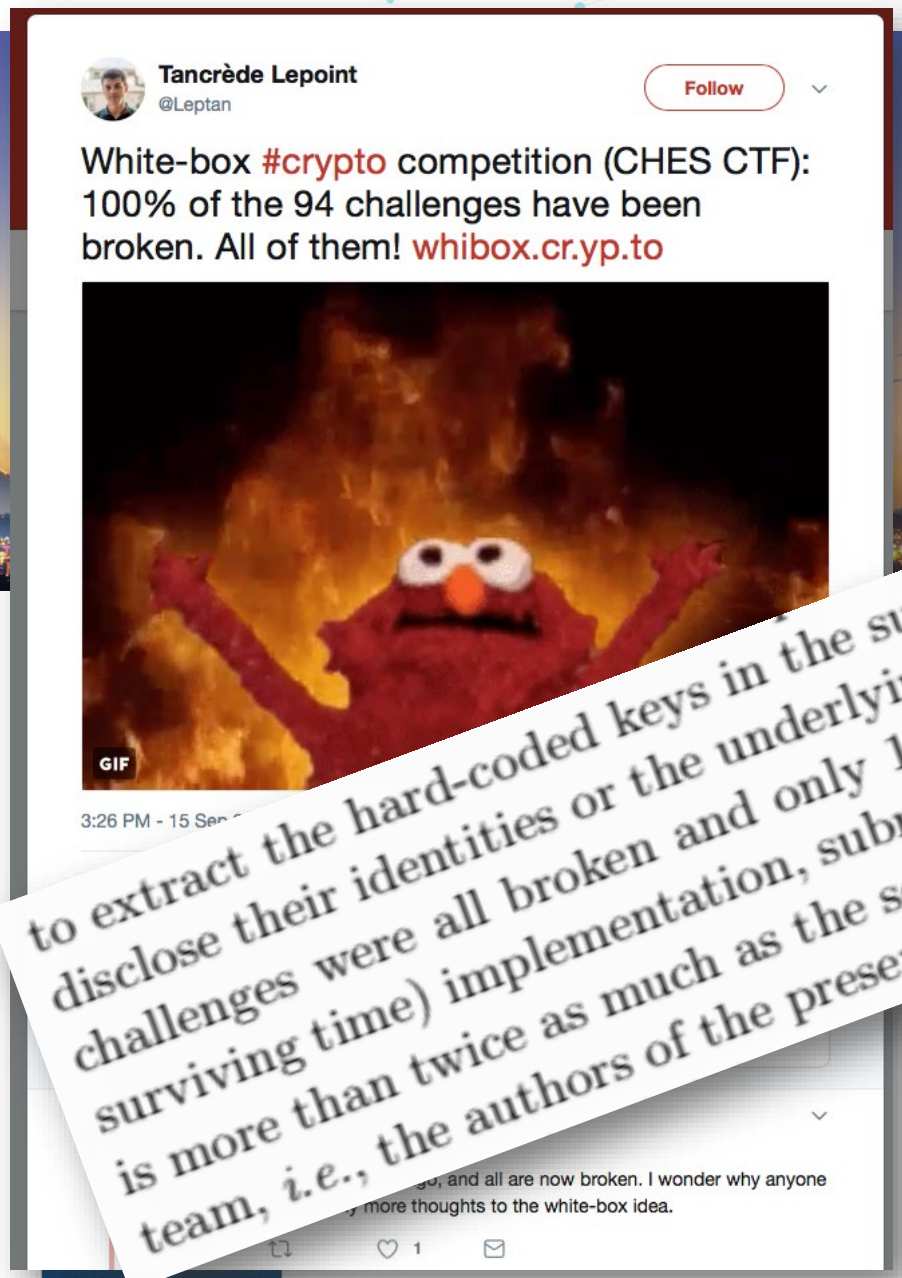
Why this competition?

The motivation for initiating the WhibOx contest comes from the growing interest of the industry towards white-box cryptography (most particularly for DRMs and mobile payments) and the obvious difficulty of designing secure solutions in a scientifically valid sense. The conjunction of these phenomena has prompted some companies to develop home-made solutions (with a security relying on the secrecy of the underlying techniques) rather than to rely on academic designs.

In such a context, the competition gives an opportunity for researchers and practitioners to confront their (secretly designed) white-box implementations to state-of-the-art attackers. It also provides attackers and evaluators with new training material.

We hope and believe that new ideas will arise from this contest and that they will have a strong, positive impact on both scientific research and industrial know-how in the field of white-box cryptography.

KATHOLIEKE UNIVERSITEIT LEUVEN



How to Reveal the Secrets of an Obscure White-Box Implementation

Louis Goubin[4], Pascal Paillier[1], Matthieu Ri
[1] C
{firstna

to extract the hard-coded keys in the submitted challenges. The participants were not expected to disclose their identities or the underlying designing/attacking techniques. In the end, 94 submitted challenges were all broken and only 13 of them held more than 1 day. The strongest (in terms of surviving time) implementation, submitted by Biryukov and Udovenko, survived for 28 days (which is more than twice as much as the second strongest implementation), and it was broken by a single team, *i.e.*, the authors of the present paper, with reverse engineering and algebraic analysis. In this

oftware implementations of
ile payment applications in which a
vironment. So far, no provably secure white-
ard, and all the published practical constructions
n analysis (DCA) and *differential fault analysis* (DFA).
ies on home-made *obscure* white-box implementations based on
ore of interest to investigate the achievable resistance of an AES im-
wart a white-box adversary in this paradigm. To this purpose, the ECRYPT
t has organized the WhibOx contest as the *catch the flag* challenge of CHES 2017.
archers and engineers were invited to participate either as designers by submitting the source code of an AES-128 white-box implementation with a freely chosen key, or as breakers by trying to extract the hard-coded keys in the submitted challenges. The participants were not expected to disclose their identities or the underlying designing/attacking techniques. In the end, 94 submitted challenges were all broken and only 13 of them held more than 1 day. The strongest (in terms of surviving time) implementation, submitted by Biryukov and Udovenko, survived for 28 days (which is more than twice as much as the second strongest implementation), and it was broken by a single team, *i.e.*, the authors of the present paper, with reverse engineering and algebraic analysis. In this paper, we give a detailed description of the different steps of our cryptanalysis. We then generalize it to an attack methodology to break further obscure white-box implementations. In particular, we formalize and generalize the *linear decoding analysis* that we use to extract the key from the encoded intermediate variables of the target challenge.

KATHOLIEKE UNIVERSITEIT LEUVEN

A New Software Approach - MPC

Paradigm

- Split the key into different random shares
- Place the random shares in different **segregated** places
- Compute without ever bringing the key together

How is this possible?

- Multi-Party Computation (MPC)

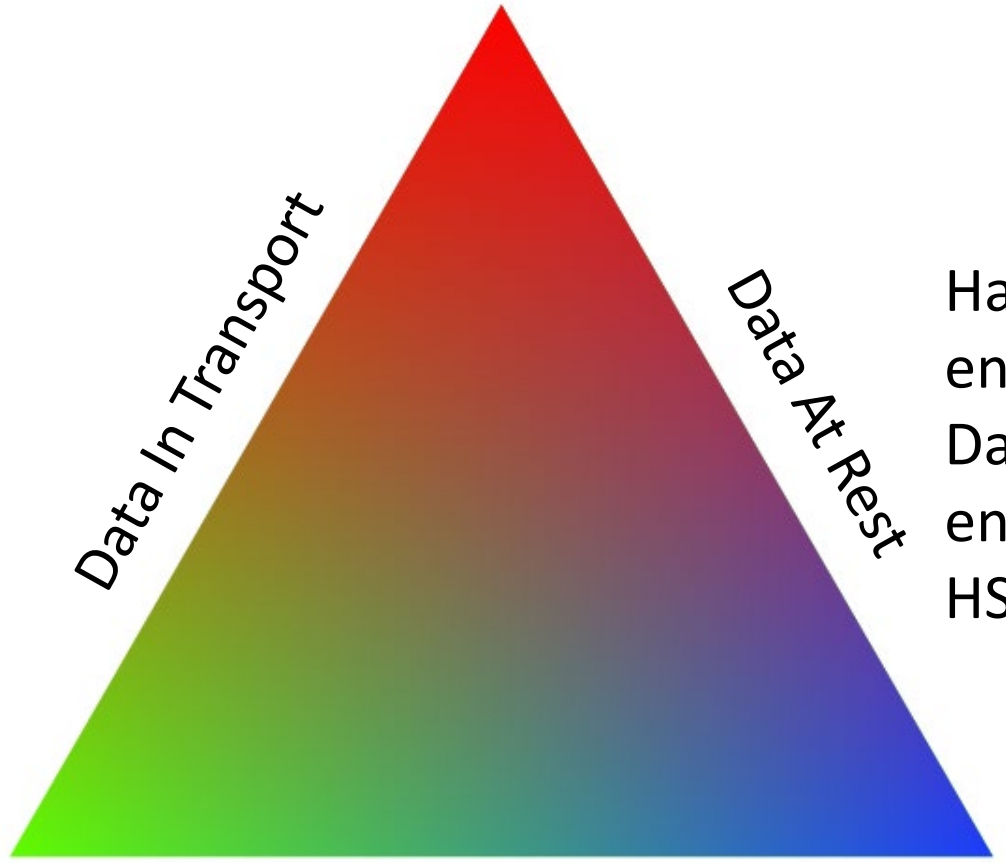
MPC: A subfield of cryptography with the goal of creating methods for parties to jointly compute a function over their inputs while keeping those inputs private

What is MPC?

MPC CAN BE USE TO COMPUTE FUNCTIONS ON DATA WHICH YOU WANT TO KEEP PRIVATE

Securing Data

TLS/SSL
IPSec

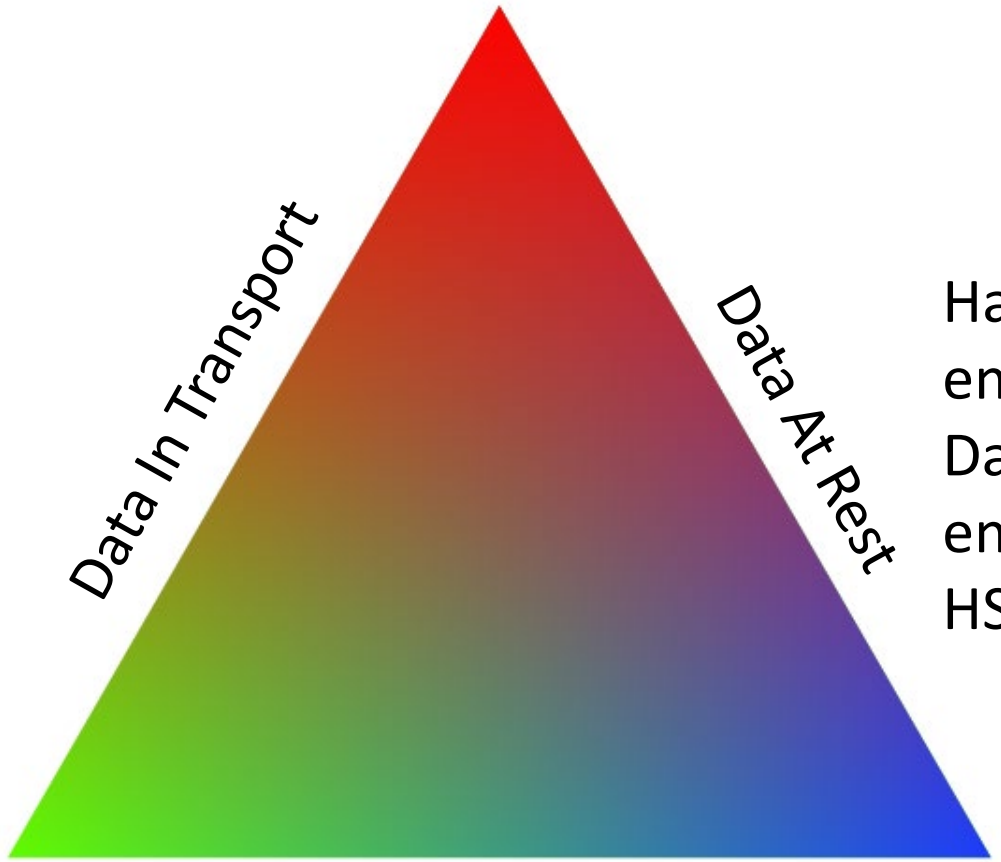


Hard disk encryption
Database encryption
HSM key storage

Data During Computation

??????????????????????????????????

Securing Data



TLS/SSL
IPSec

Hard disk encryption
Database encryption
HSM key storage

Voting

Citizen Privacy

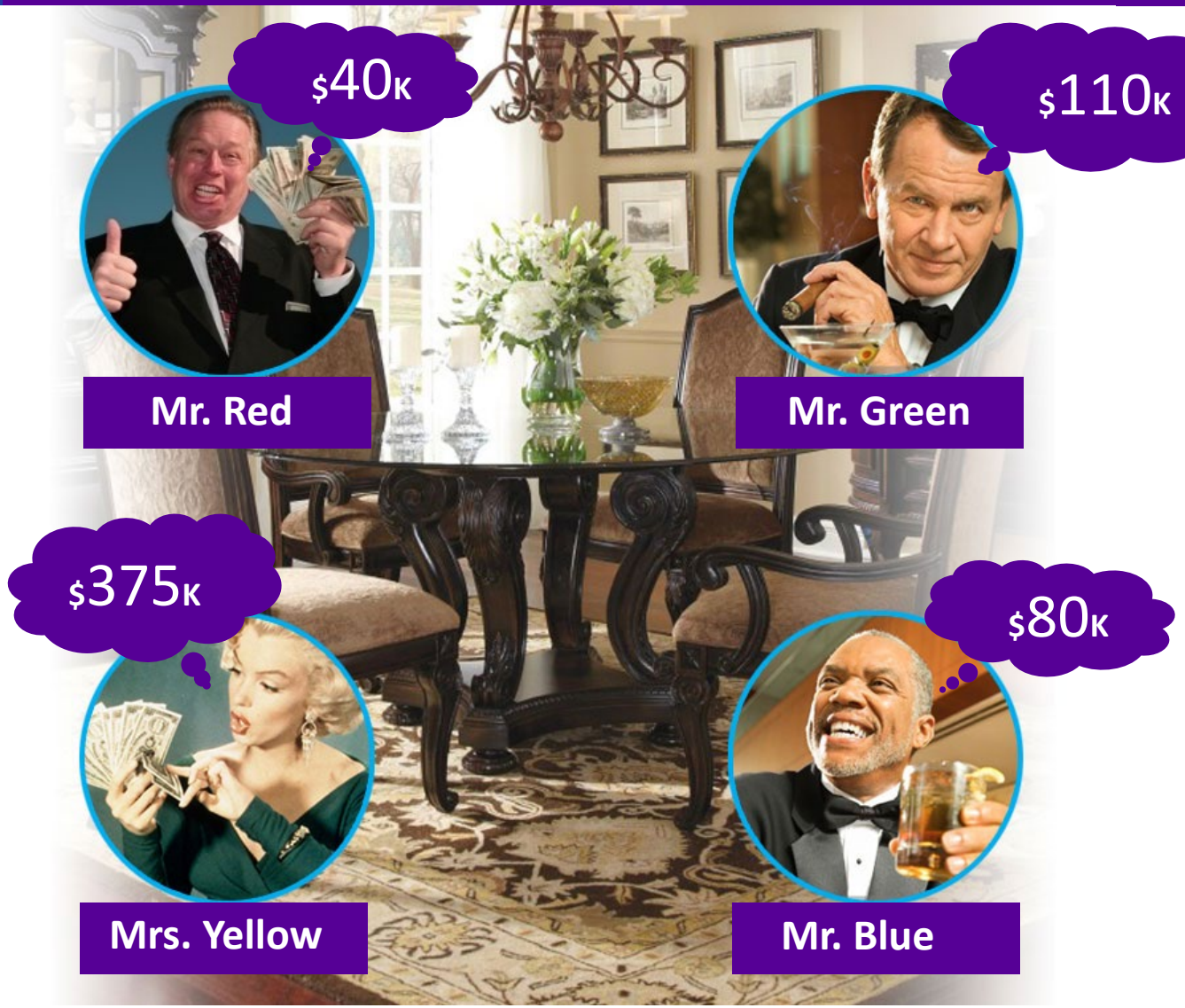
GDPR

Genomics

Public Policy

KATHOLIEKE UNIVERSITEIT LEUVEN

Dining Bankers: The Problem



A group of bankers go to lunch.

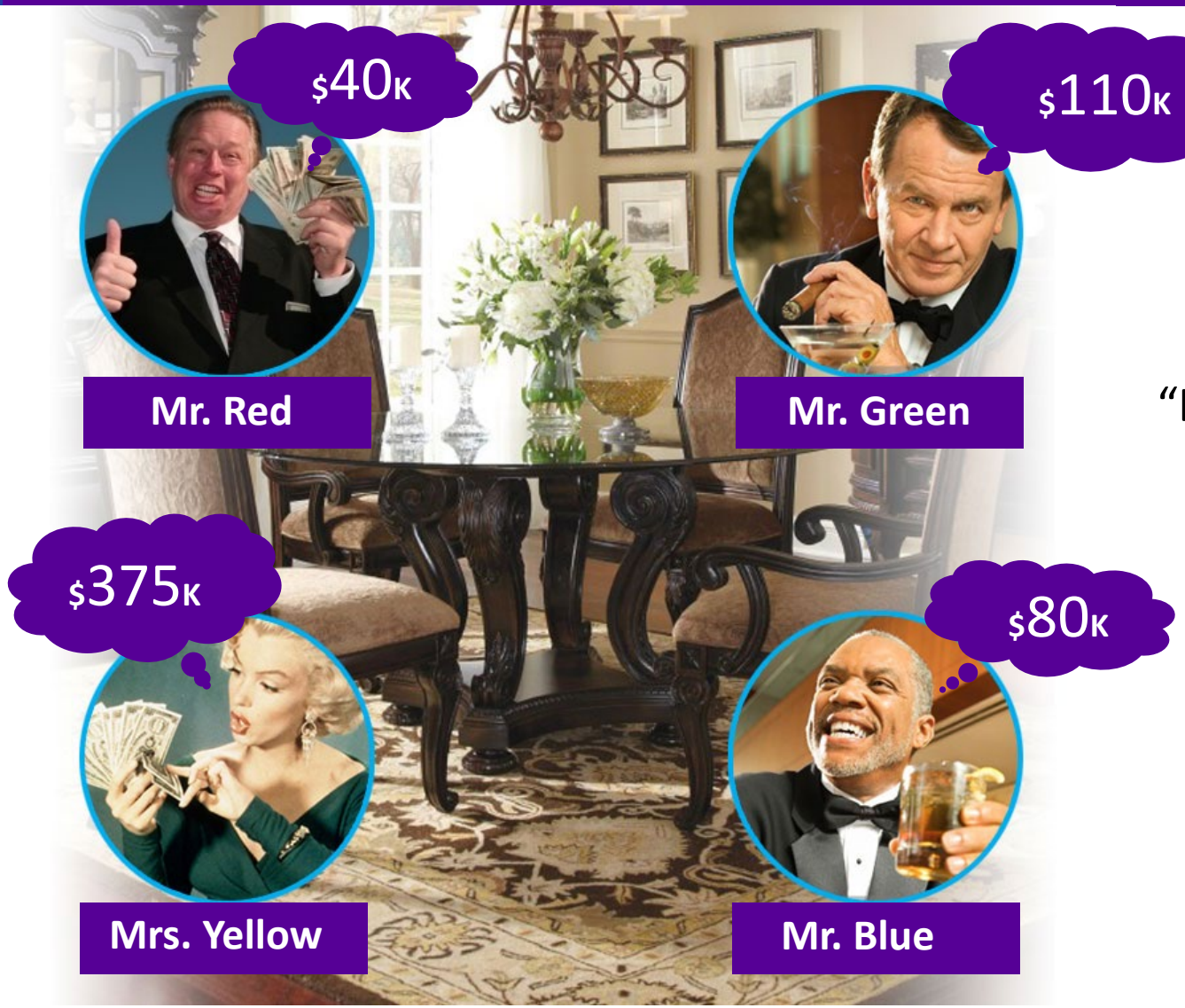
They are celebrating their bonuses just being paid.

Each has been given a bonus of x_i dollars.

The one with the biggest bonus should pay.

But they do not want to reveal their bonus values.

Dining Bankers: The Problem [2]



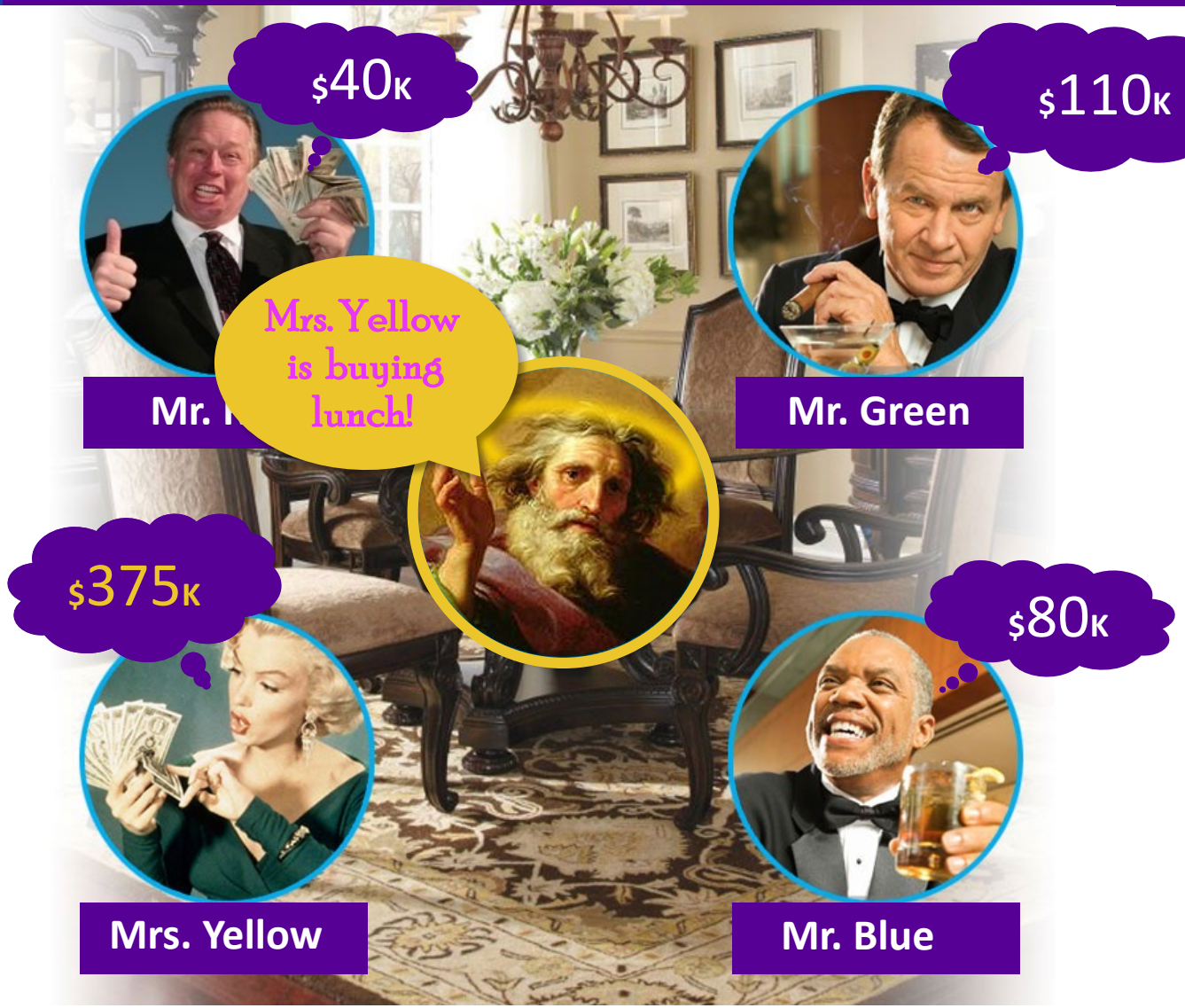
What they want to compute is the function

"Banker name with $\max(x_{red}, x_{green}, x_{blue}, x_{yellow})$"

without revealing the x_i values.

This problem (Millionaires Problem) introduced by Andy Yao in early 1980s. Yao won the Turing Award for this and other work.

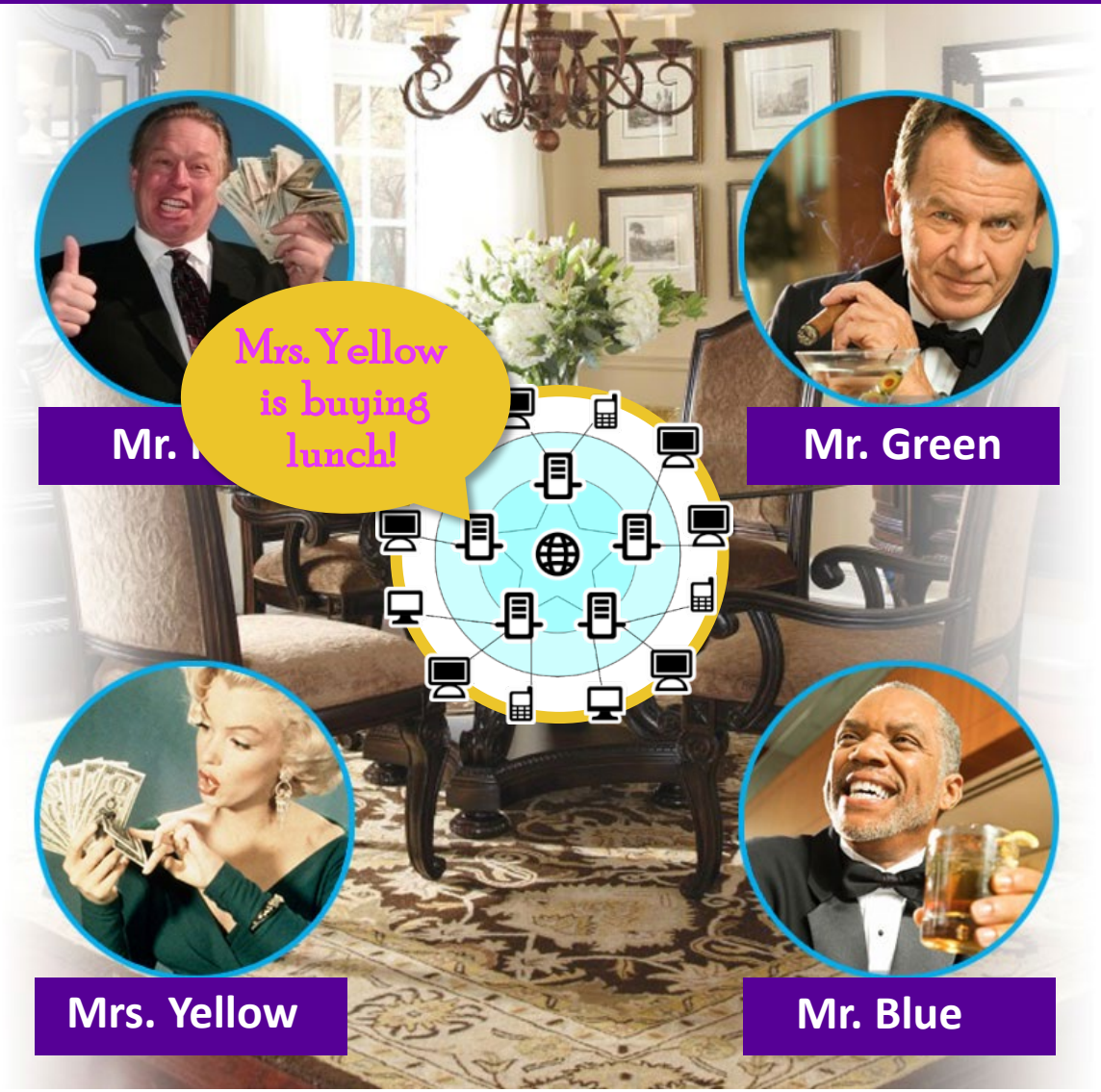
Dining Bankers: Naïve Solution



If the bankers had a person they trusted they could get this person to compute the answer to their problem for them.

They give the trusted person their bonus values and the trusted person computes who should pay for lunch.

Dining Banker: MPC Solution



In real life such trusted people do not exist, or are hard to come by. So we want **a protocol to compute the function securely.** This is what MPC does.

It emulates a trusted party, enabling mutually distrusting parties to compute an arbitrary function on their inputs.

All that is revealed is what can be computed from the final output.

Security of MPC Protocols

- **Security guaranteed even for malicious adversaries**
  - Adversary who breaches machine and can run arbitrary attack code
  - Adversary knows protocols and is “expert” cryptographer
- **Number of parties and corruptions**
  - Honest majority vs dishonest majority
  - 2 parties, 3 parties, many parties
- **Latency**
  - Protocols with a few rounds of communication (guaranteeing low latency) vs many rounds of communication

Security of MPC

Mathematically proven guarantee of security

- If attacker does not get to both machines (in one refresh period) nothing can be learned
- Clear security model – not cat-and-mouse
- No side-channels by design

Achieving a high level of security – it's all about separation

- Different administrators (or at least different credentials)
- Different OSs or defenses
- Different environments
  - Hybrid cloud, different clouds, endpoint/cloud

Key Usage Mitigation with MPC

- Key theft is always a catastrophe
  - Attacker has unlimited access to key operations
- Mass key usage mitigation (e.g., decrypt credit card numbers)
  - Run anomaly detection on MPC machines
- Limited key usage is also a problem
  - Anomalous usage can be detected for mass use by an attacker
  - Low usage cannot be detected and prevented
- **Examples**
  - Cryptocurrency usage
  - Code signing

Key Usage Mitigation with MPC

- **MPC supports authorization structures**
  - Can define arbitrary sets of servers/clients
  - In each set, can define subset size needed to authorize
  - Can define logical AND/OR between sets
  - Enforcement is **cryptographic** (not access control)
  - Useful for highvalue signing cases such as code-signing or cryptcurrencies

- **Can also protect client certificates**
  - Client authenticates to HSM/MPC cluster
  - Use MPC to protect client authentication key

Security Properties of MPC

- Mathematical proof of security (assume no access to enough shares)
- Strong separation achieves high level of security
  - Passive MPC server can run only MPC API
    - Makes it equivalent to dedicated hardware, and as protected as HSM from software attacks
  - Can place in separate physical locations for physical protection
    - Consider mobile and server, different countries for subpoenas
- Has all benefits of software
  - Crypto and vulnerability agility
  - Management ease
  - Relevant for endpoint and server

Comparison of Approaches

What Metrics Should We Look At For Comparison?

- **Security properties**
  - Key extractability – software attack
  - Key extractability – physical access
  - Crypto and vulnerability agility
  - Key usage mitigation
- **Trust level for security properties**
  - Experience (for good and bad)
  - Mathematically proven properties
- **Usability**
  - Platform agnostic
  - Functional flexibility
  - Performance
  - Administration

HW and SW Methods: Detailed Comparison

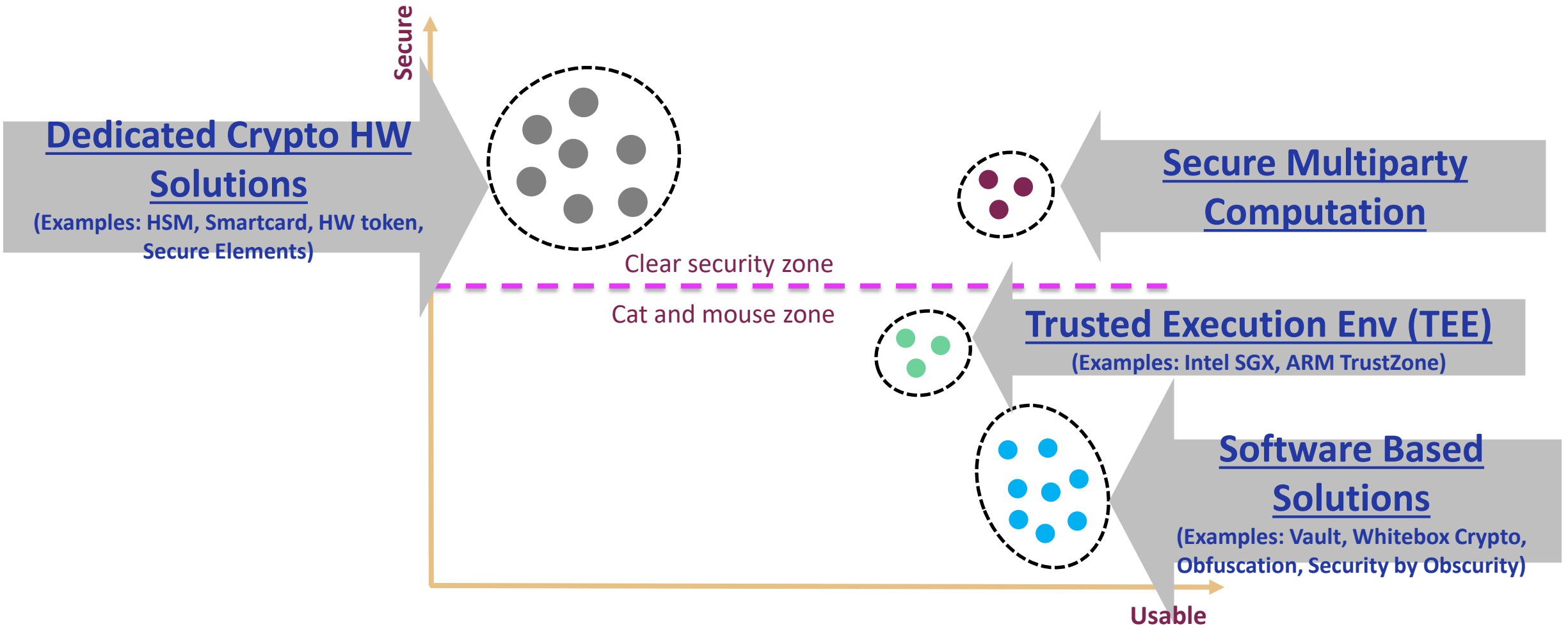
		Software (e.g. whitebox crypto, obfuscation)	TEE (e,g. Intel SGX, ARM TrustZone)	Dedicated Crypto Hardware (e.g. HSM, Smartcard)	MPC
Security	Key extractability – software access	✗ Exposed to co-location or process-injection attacks	✗ Exposed to co-location attacks (using SW-based side channel)	✓ Cannot run general software (assuming no vulnerability)	✓ Need to breach both machines simultaneously
	Key extractability – physical access	✗	✓ Manufacturers guarantee nothing	✓ Need physical access and significant effort	✓ Need physical access to both; can separate
	Agility	✓	Partial[1]	✗	✓
	Key usage mitigation	N/A		✗	✓ Anomaly detection, authorization structures and client cert protection
Usability	Platform agnostic	✓	Endpoint: Partial Server: Limited, unclear outlook	✗	✓
	Functional flexibility	✓	✓	✗	✓
	Performance	✓	✓	Efficient for master key operations only	Efficient for master key operations only
	Administration	✓	✓	✗	✓

KATHOLIEKE UNIVERSITEIT LEUVEN

RSAConference2019

1. Internal SGX architecture uses non-PQC secure algorithms such as AES-128, requiring major firmware/hardware upgrade to comply with PQC.

HW and SW Methods for Key Protection



Summary

Dedicated crypto hardware

- Key extractability is very hard
- Security of other properties is lower (agility, key usage mitigation)
- Scores low on administration and flexibility (e.g., cloud usages, endpoint)
- Performance good for master-key operations
  - Not suitable for mass symmetric encryptions

Trusted execution enclaves

- Key extractability is very possible
  - Side channels are very hard to prevent, if not impossible
  - Many successful demonstrations of attack in recent years
- Security of other properties is mixed (agility, key usage mitigation)
- Scores reasonably on administration and flexibility (e.g., cloud usages, endpoint)
- Excellent performance

Software-based solutions

- Key extractability is easy, and so very low security
- Lack of formal security guarantees for practical solutions
- Methods with formal security guarantees are utterly impractical
- Scores well on all other aspects

MPC based solutions

- Key extractability is very hard – mathematical proof of security (assumes good separation)
  - Different to dedicated crypto hardware – not an exact comparison
- Provides full agility and different key usage mitigation techniques
- Scores high on administration and flexibility (e.g., cloud usages, endpoint)
- Performance good for master-key operations
  - Not suitable for mass symmetric encryptions

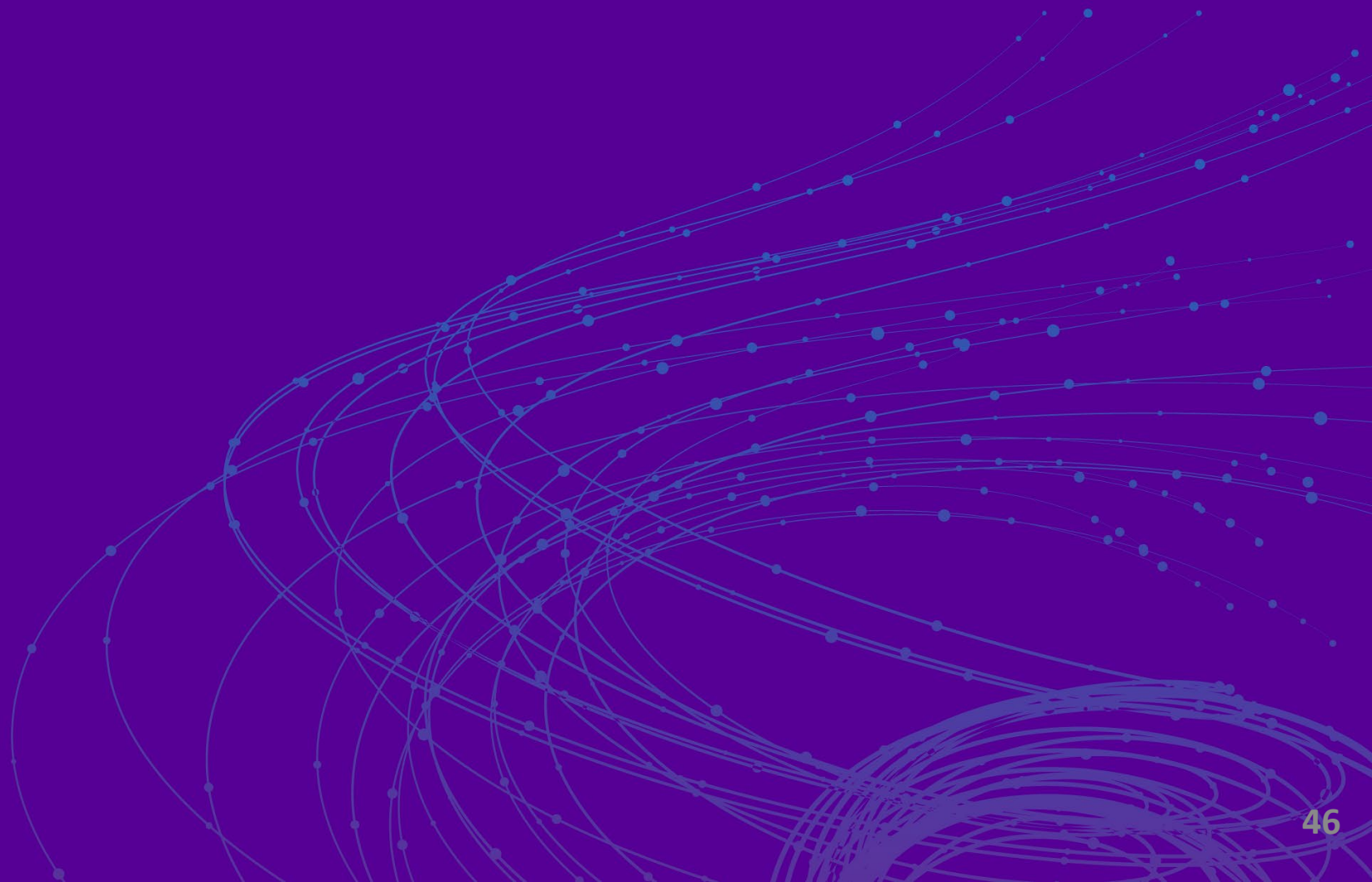
Apply What We Have Covered

Application of this

- Work out in your organization what are the requirements for key protection
  - Do you need HSMs due to PCI compliance?
  - What applications can be utilize non-HSM controls?
  - How easy is it to manage your infrastructure?
  - Do you require strong coupling between complex authorizations and key usage?
    - e.g. as in cryptocurrencies or code-signing
  - Are your solutions susceptible to side-channel attacks?
  - How easy is it to spot anomalies?
  - Are you crypto-agile against new threats?
  - Can you easily switch algorithms when vulnerabilities are found?
  - How is key life cycle managed (creation, usage, rolling over and destruction of keys)

RSAConference2019

Questions?



RSAConference2019

Back Up Slides

MPC Example - Securely Computing the RSA Function

RSA signing and decryption:

- Private key: (d, N)
- Public key: (e, N)
- Private operation (sign/decrypt): $z = y^d \bmod N$

RSA key sharing

- Server S_1 has a random d_1
- Server S_2 has $d_2 = d - d_1 \bmod \phi(N)$
- Note that $d_1 + d_2 = d \bmod \phi(N)$
- Security:
  - d_1 reveals nothing about d since it's random
  - d_2 reveals nothing about d_1 completely hides d

MPC Example - Securely Computing the RSA Function

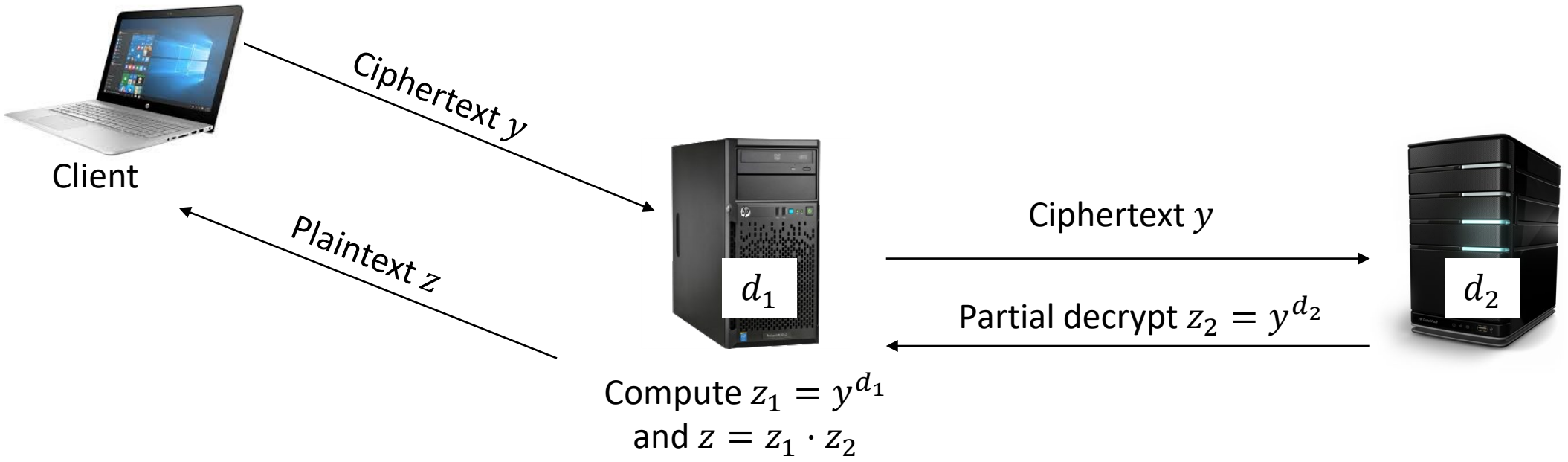
Recall: server S_1 has d_1 and server S_2 has d_2, such that $d_1 + d_2 = d$

Securely computing the private operation $x = y^d \bmod N$

- Server S_2 computes $z_2 = y^{d_2} \bmod N$ and sends to server S_1
- Server S_1 computes $z_1 = y^{d_1} \bmod N$
- Server S_1 computes $z = z_1 \cdot z_2 \bmod N$
- Server S_1 *verifies the result* by checking that $y = z^e \bmod N$
- Note: $z = z_1 \cdot z_2 = y^{d_1} \cdot y^{d_2} = y^{d_1+d_2} = y^d \bmod N$
  - The last equality holds since addition in exponent is mod $\phi(N)$

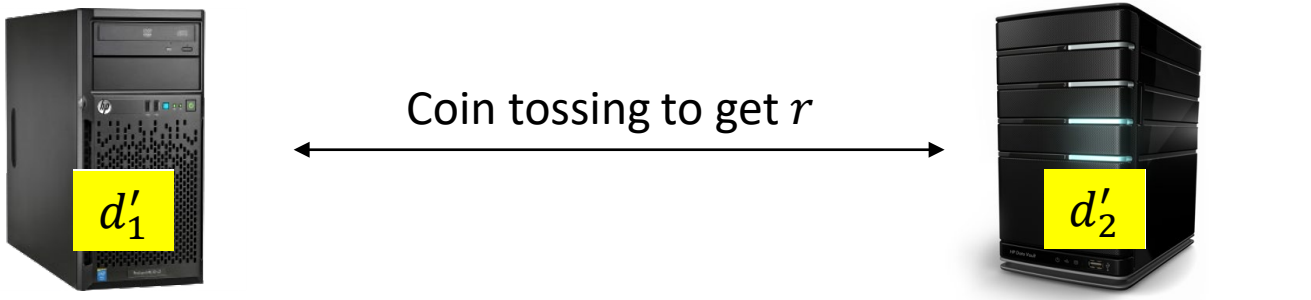
Two-Party Protocol for RSA Decrypt/Sign

Inputs: Server S_1 has key share d_1 and ciphertext y, and server S_2 has key share d_2



RSA Secret Share Refresh

- At fixed intervals (e.g., every hour), sharing of secret is refreshed
- For RSA:



Compute $d'_1 = d_1 + r$

Compute $d'_2 = d_2 - r$

Note that given d_1 and $d'_2 = d_2 - r$, nothing can be learned about d