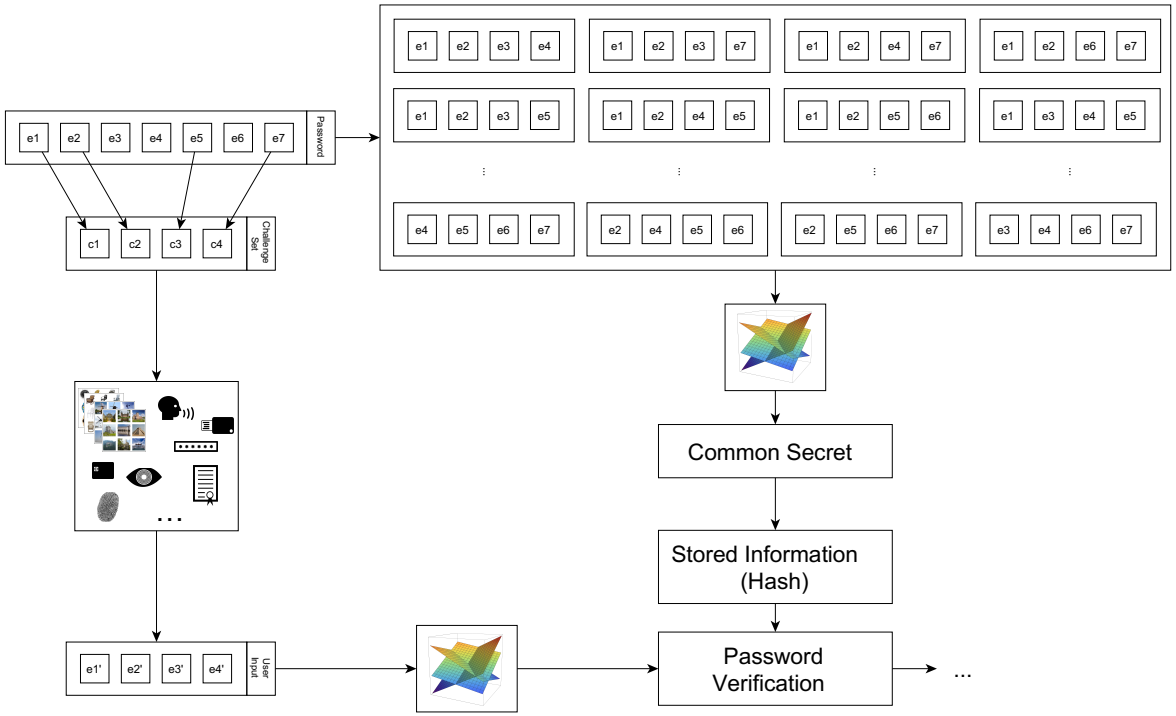
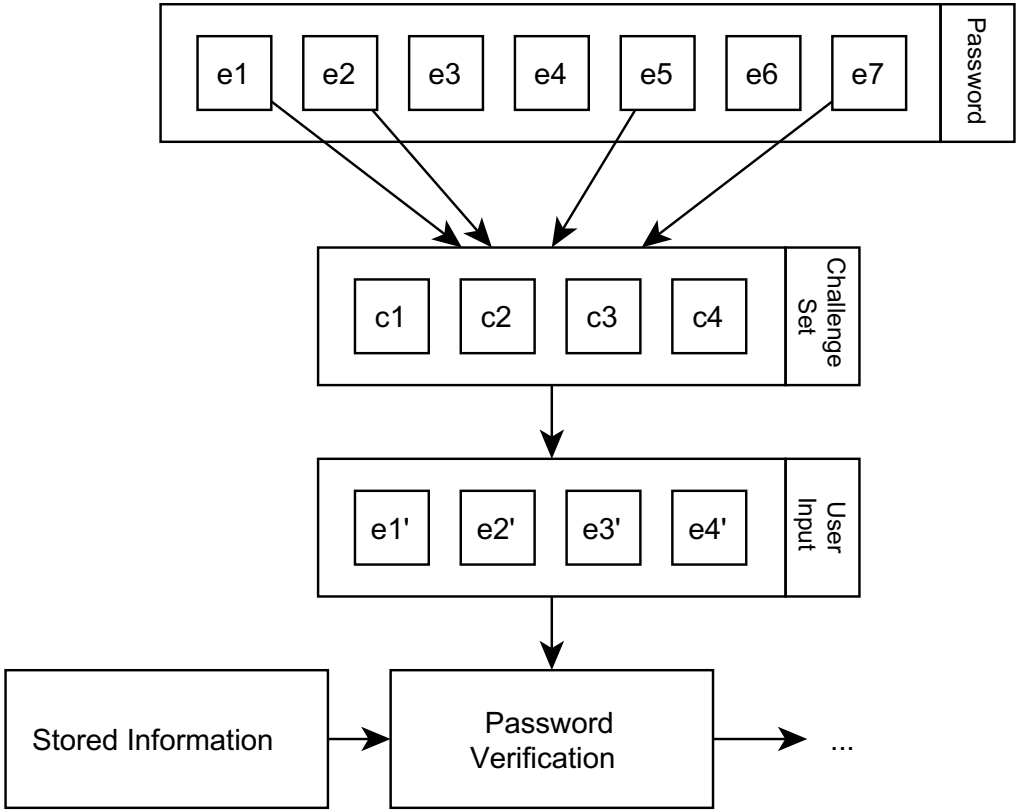
Secure and Efficient Key Derivation in Portfolio Authentication Schemes Using Blakley Secret Sharing

Peter Mayer, Melanie Volkamer

TECHNISCHE UNIVERSITÄT DARMSTADT
SECUSO SECURITY · USABILITY · SOCIETY
CASED Usable Security Lab Crypto Lab



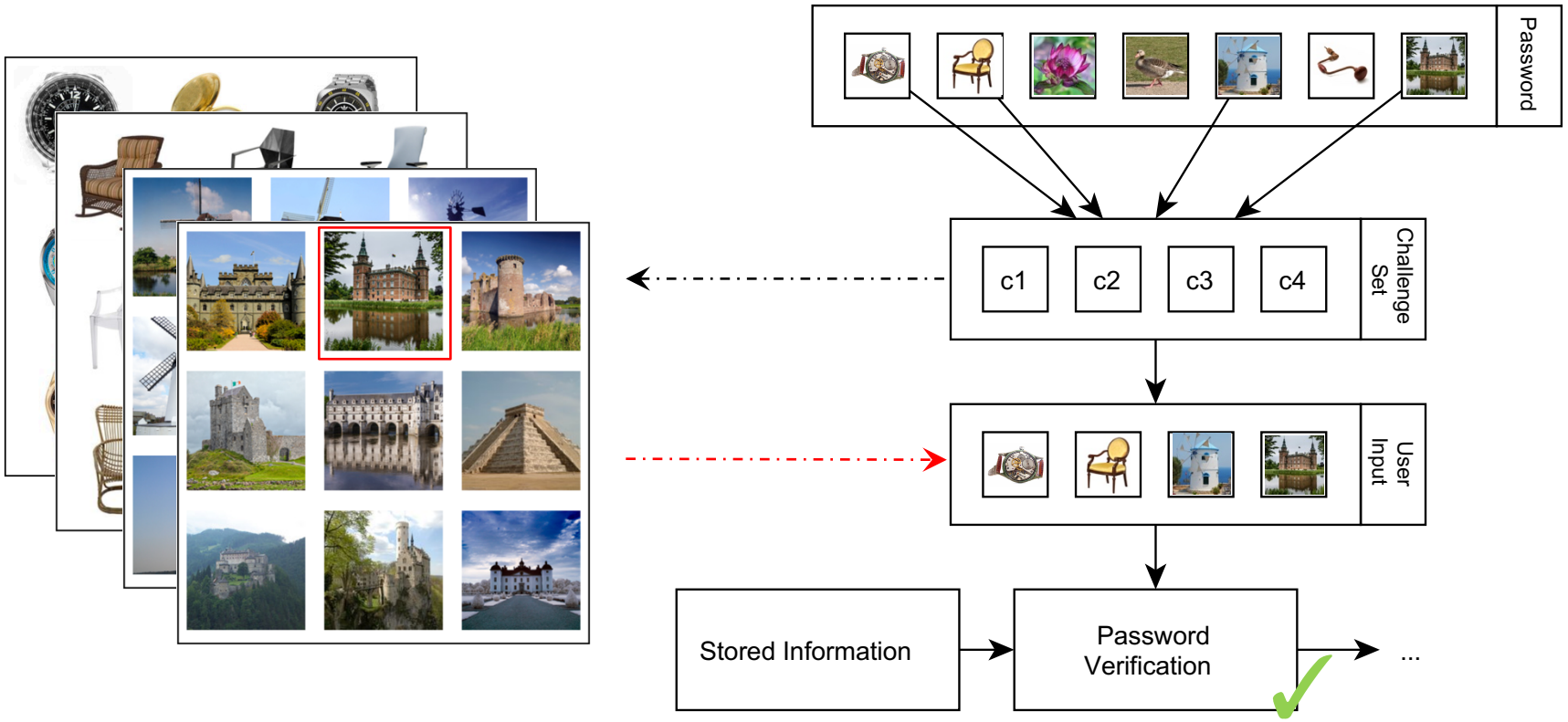
Portfolio authentication



Why portfolio authentication?

<shoulder surfing image>

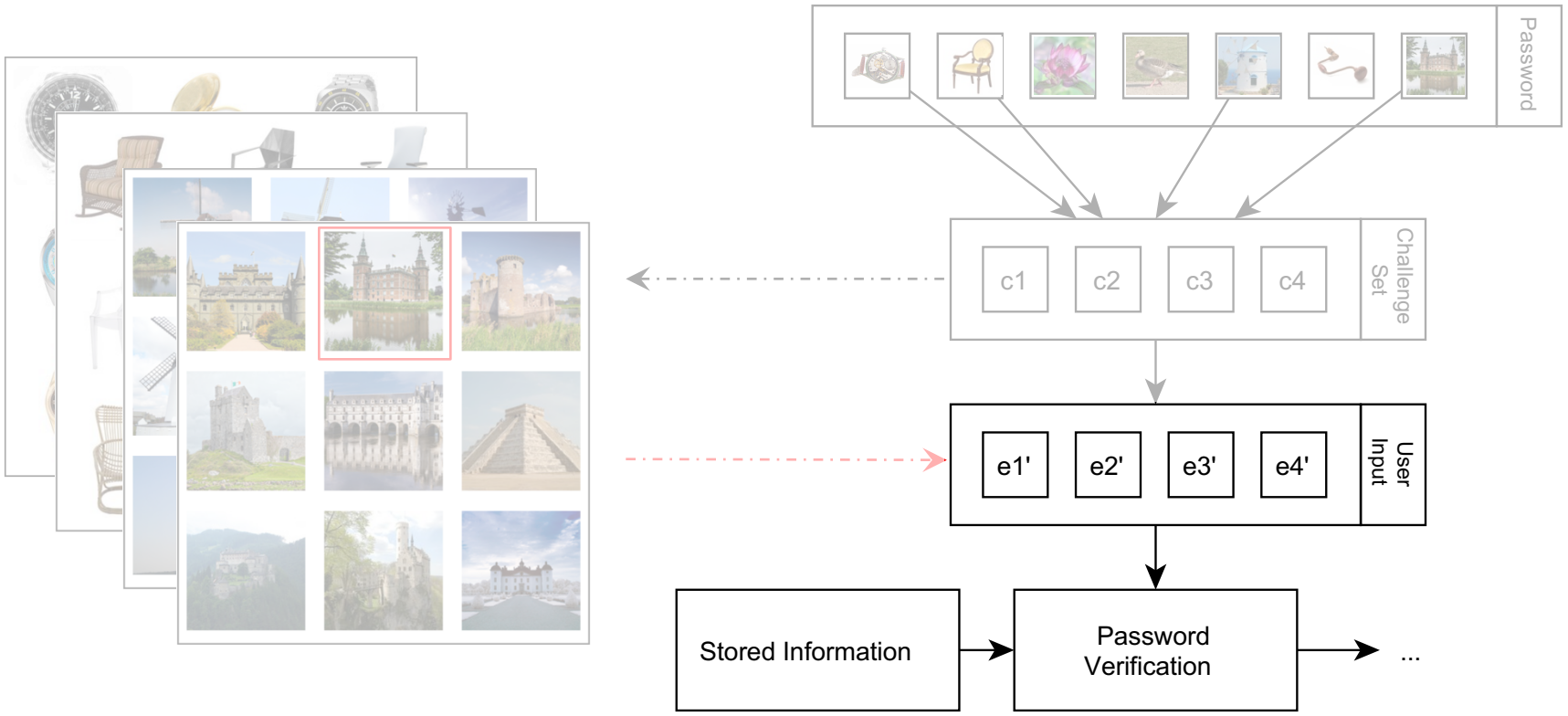
Graphical recognition-based portfolio authentication



TECHNISCHE UNIVERSITÄT DARMSTADT

SECUSO SECURITY · USABILITY · SOCIETY

Graphical recognition-based portfolio authentication



Portfolio authentication

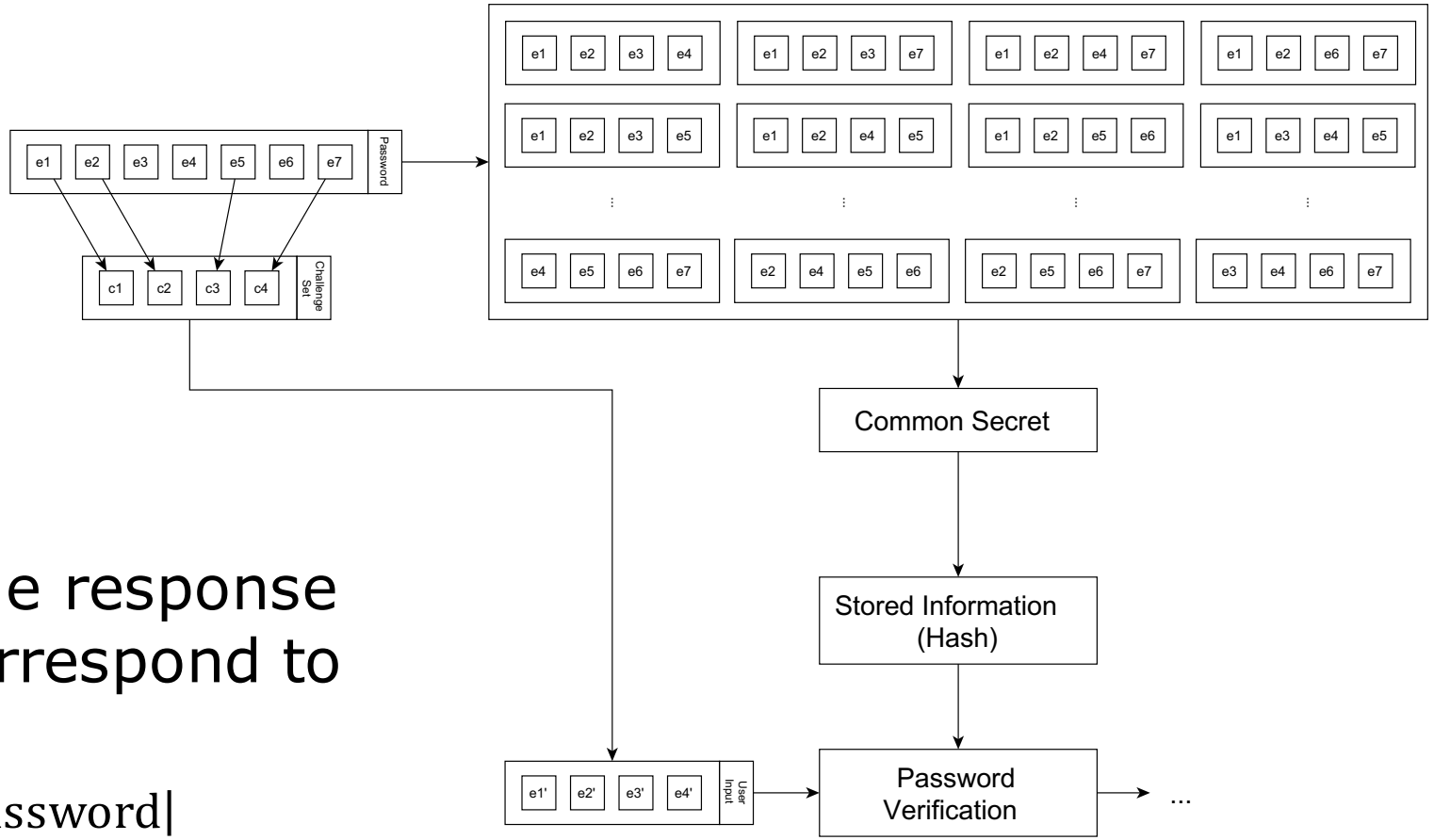
- Naïve approach
  - Generate & store the hashes for all authorized subsets
  - Check if user input matches one of the hashes

  - Easy to implement

  - Factorial growth of number of hashes

    ➔ inefficient

(t,n)-threshold verification



challenge response pairs correspond to parties

- ➔ $n = |\text{password}|$
- ➔ $t = |\text{authorized subsets}|$

TECHNISCHE UNIVERSITÄT DARMSTADT

SECUSO SECURITY · USABILITY · SOCIETY

Secret sharing

Usual Secret sharing

1. Choose construct
2. Choose common secret
3. Generate and distribute shares
4. Combine shares to reconstruct common secret

→

Shamir secret sharing

1. Dealer chooses polynomial *a* of degree $t-1$
2. the common secret is a_0
3. Choose shares $y_i = a(x_i)$
4. Reconstruct a_0 using Lagrange interpolation

Secret sharing

Usual Secret sharing

1. Choose construct
2. Choose common s
3. [illegible]
4. [illegible] shares to reconstruct common secret

Shamir secret sharing

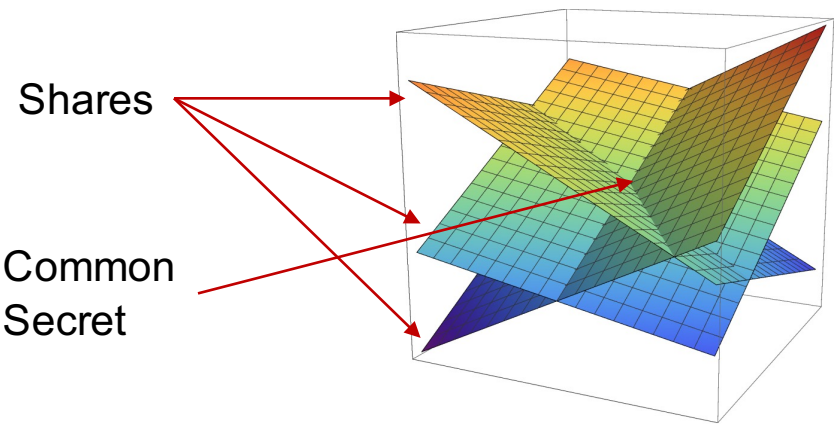
1. Dealer ch[illegible]
2. [illegible] as the common secret
3. Choose shares $y_i = a(x_i)$
4. Reconstruct a_0 using Lagrange interpolation

Problem: in the verification scenario the shares are already predetermined by the password

Blakley secret sharing

Blakley secret sharing

- Uses hyperplane geometry in t-dimensional space



1. Common secret is the first coordinate of random t-dimensional point x in $GF(p)$
2. The dealer chooses t values m_{ij} at random for each party…
3. …and calculates the shares y_i for all parties $m_{i1}x_1 + m_{i2}x_2 + \cdots + m_{it}x_t = y_i$
4. To reconstruct the secret, t parties combine their shares to solve $M'x = y'$

Blakley secret sharing

- As is, the problem persists
- but it is possible to adapt Blakley secret sharing allowing preselection of the shares

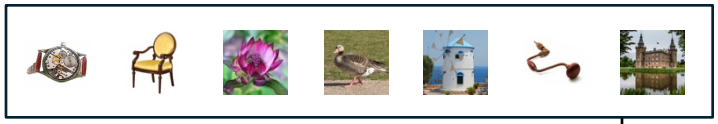
1. Common secret is the first coordinate of random t-dimensional point x in $GF(p)$
2. The dealer chooses t values m_{ij} at random for each party…
3. …and calculates the shares y_i for all parties $m_{i1} + m_{i2} + \cdots + m_{it} = y_i$
4. To reconstruct the secret, t parties combine their shares to solve $M'x = y$

(t,n)-threshold verification

Procedure

Enrollment

1. Choose prime p and random t-dimensional point x in $GF(p)$
2. Derive share $y_i = KDF(e_i)$ from password elements e_i
3. Choose coefficients $m_{i1}, \dots, m_{it-1}$
4. Solve $m_{i1} + m_{i2} + \cdots + m_{it} = y_i$ for m_{it}
5. Store M and s



$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_t \end{pmatrix}$$

$$s = KDF(x_1)$$

$$KDF(e_i)$$

$$y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

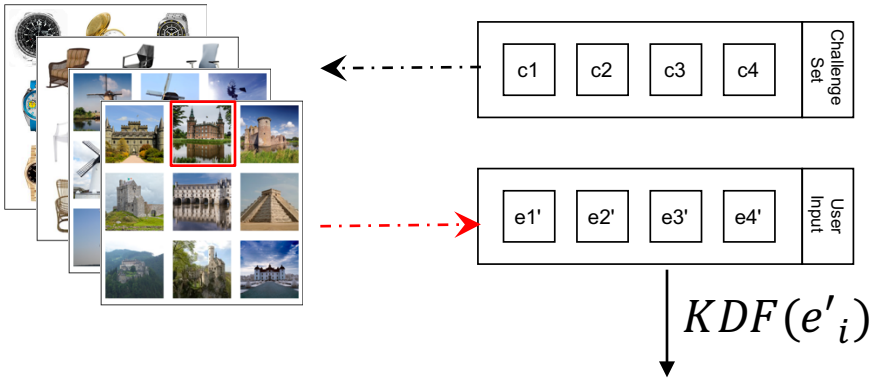
$$\begin{pmatrix} m_{11} & \cdots & m_{1t} \\ \vdots & \ddots & \vdots \\ m_{n1} & \cdots & m_{nt} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_t \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

(t,n)-threshold verification

Procedure

Verification

1. Gather user input e'_i
2. Derive shares y'_i from the user input e'_i
3. Build M from the m_{ij} corresponding to the challenges
4. Solve $M'x = y'$ for x
5. Check whether $KDF(x_1) \stackrel{?}{=} s$



$$y' = \begin{pmatrix} y'_1 \\ \vdots \\ y'_t \end{pmatrix}$$

$$\begin{pmatrix} m'_{11} & \cdots & m'_{1t} \\ \vdots & \ddots & \vdots \\ m'_{t1} & \cdots & m'_{tt} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_t \end{pmatrix} = \begin{pmatrix} y'_1 \\ \vdots \\ y'_n \end{pmatrix}$$

$$s \stackrel{?}{=} KDF(x_1)$$

Security properties

Guessing Resistance

- All variables in $GF(p)$
- p distinct values
  - on average $\frac{p}{2}$ attempts
- Perfect secret sharing
- To ensure strength H:

$$p \geq 2^{H+1}$$

Secure Storage

- Security of storage completely depends on used KDF
- If KDF is secure, (t,n)-threshold verification is secure

Efficiency properties

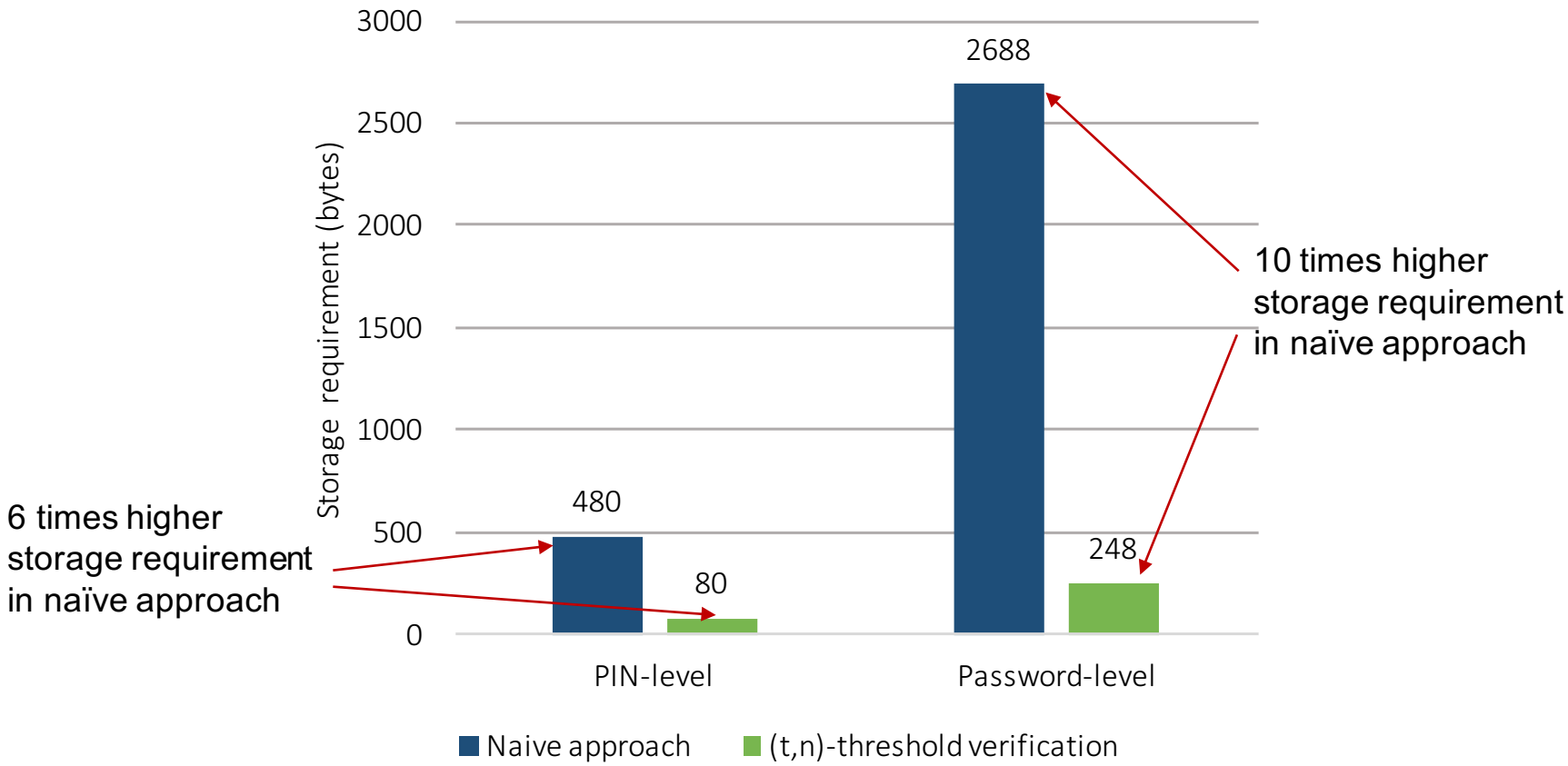
- Evaluation against naïve approach
  - Only alternative available
- Storage
- Computation time

Storage comparison

Two security levels

- PIN-level (password space of 10^4 entries)
  - Example: 4 out of 6, alphabet: 0-9
- Password-level (password space of $\sim 2^{27}$ entries)
  - Example: 6 out of 9, alphabet: 95 keys on US standard keyboard

Storage comparison



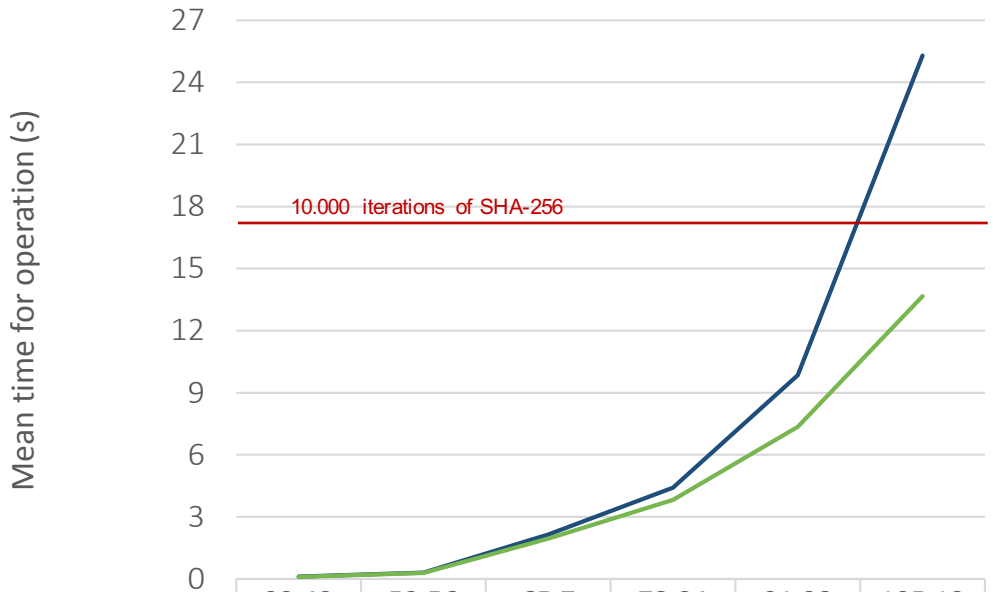
TECHNISCHE UNIVERSITÄT DARMSTADT

SECUSO SECURITY · USABILITY · SOCIETY

Computation time

- Monte Carlo evaluation
  - 10.000 iterations
- Implementation in Mathematica
  - Salted SHA-256 as KDF
  - Code available on GitHub
    - https://github.com/SecUSo/t-n-threshold-verification

Computation time comparison



	39,42	52,56	65,7	78,84	91,98	105,12
Creation	0,113	0,310	2,142	4,419	9,843	25,300
Verification	0,101	0,283	1,954	3,828	7,360	13,666

Strength of authentication mechanism *H*

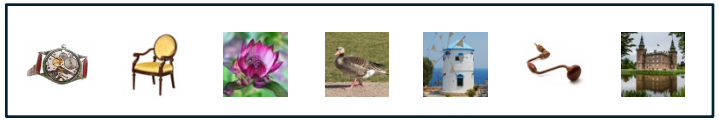
All timings are smaller than the usual 10.000 iterations of SHA-256 recommended for password storage on the same platform.

Summary

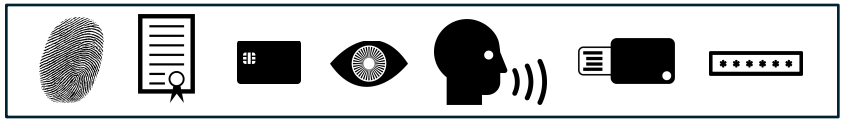
- (t,n)-threshold verification
  - Enabler for new authentication technologies
  - More efficient in terms of storage and time
  - Allows secure & efficient storage in portfolio authentication schemes for the first time
  - Allows additional operations to easily modify the password

What’s next?

- Portfolio authentication originally conceived for graphical recognition-based schemes

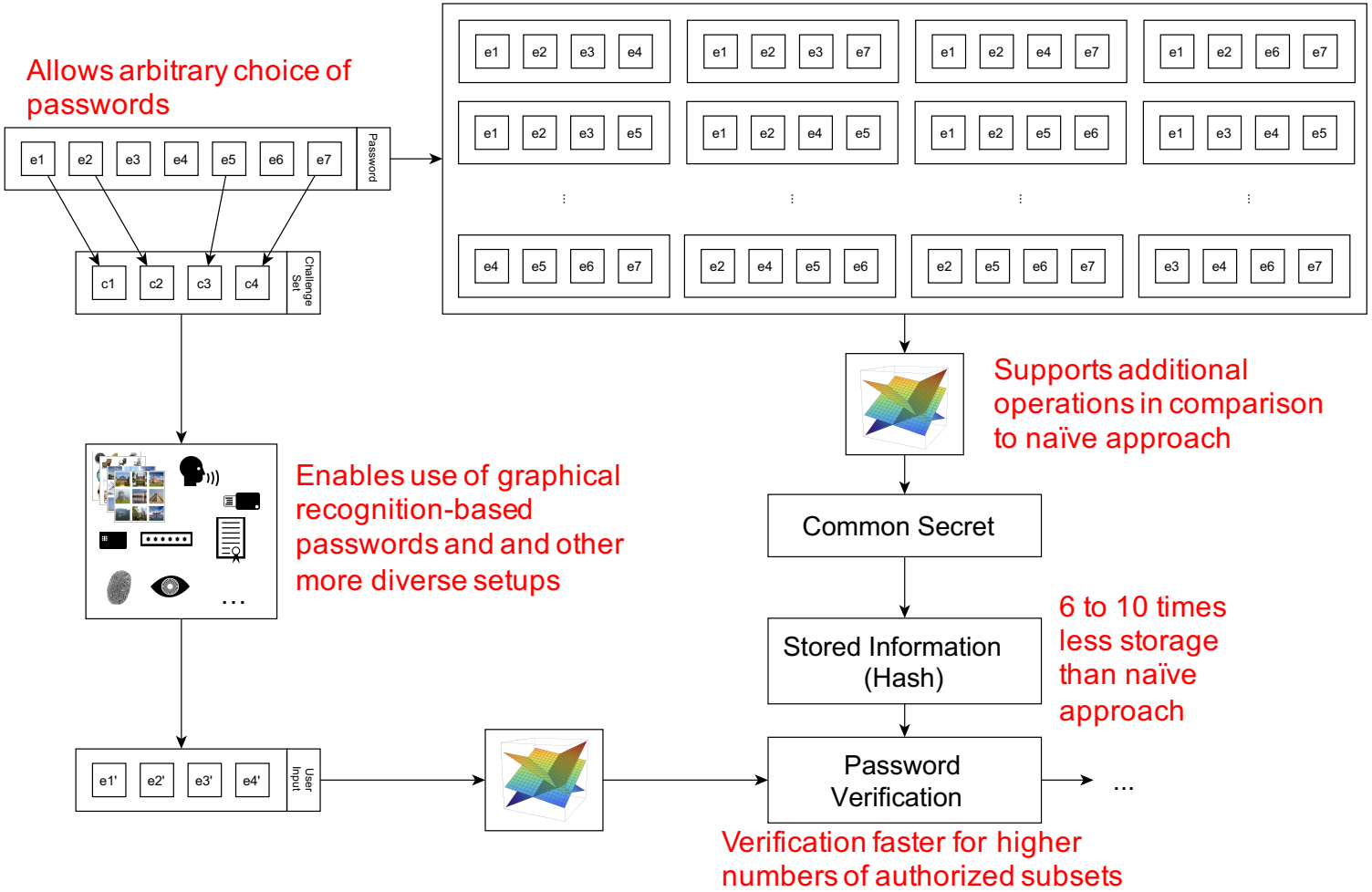


- Applications go far beyond this limited scenario



Images by Andrew Forrester, Wilson Joseph, Ricardo Moreira, Laurent Canivet, Magicon, Edward Boatman and Evgeniy Kozachenko

(t,n)-threshold verification



TECHNISCHE UNIVERSITÄT DARMSTADT

SECUSO SECURITY · USABILITY · SOCIETY