Secure authentication and communication for embedded systems using a PUF/TRNG combined circuit

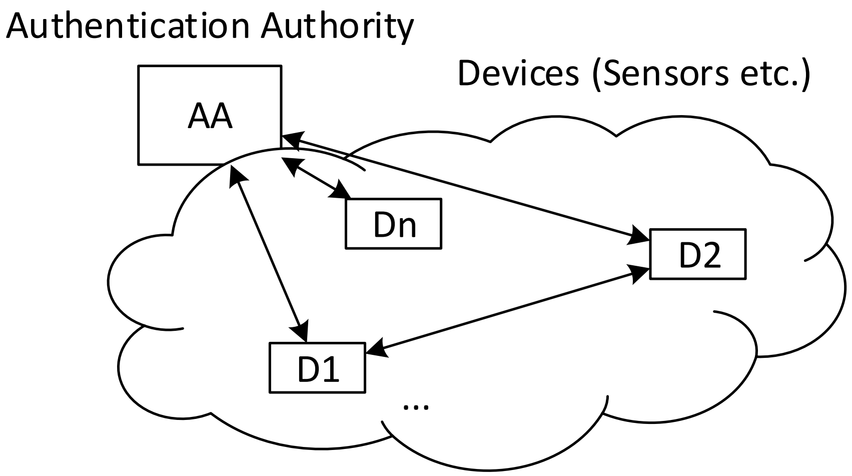
Simona Buchovecká, Róbert Lórencz, Jiří Buček and Filip Kodýtek

Czech Technical University in Prague
Faculty of Information Technology

Outline

Motivation and Goals

- Secure and simple communication and authentication for embedded and other simple interconnected devices
- Secure and simple storage of generated keys on embedded devices to simplify key management



Background and State-of-the-Art (1)

- Generally accepted requirements on key generation
  - A source of true randomness for unpredictable and unique fresh keys
  - A protected memory which reliably stores the keys information while shielding it completely from unauthorized parties
- TRNGs mostly used for key generation nowadays
  - Quality of TRNG is crucial
  - Secure storage of generated keys is crucial
- In last years, increasing trends in PUF usage as another approach for key generation
  - Challenges in slightly variable PUF output

Background and State-of-the-Art (2)

- A lot of work done in the areas of TRNGs (and PUFs) designs with various designs being proposed

- Systematic and formalized approach to key management in hardware devices and embedded systems with properly defined requirements, as well as efficient light-weight modules for key generation and storage and secure usage are still missing

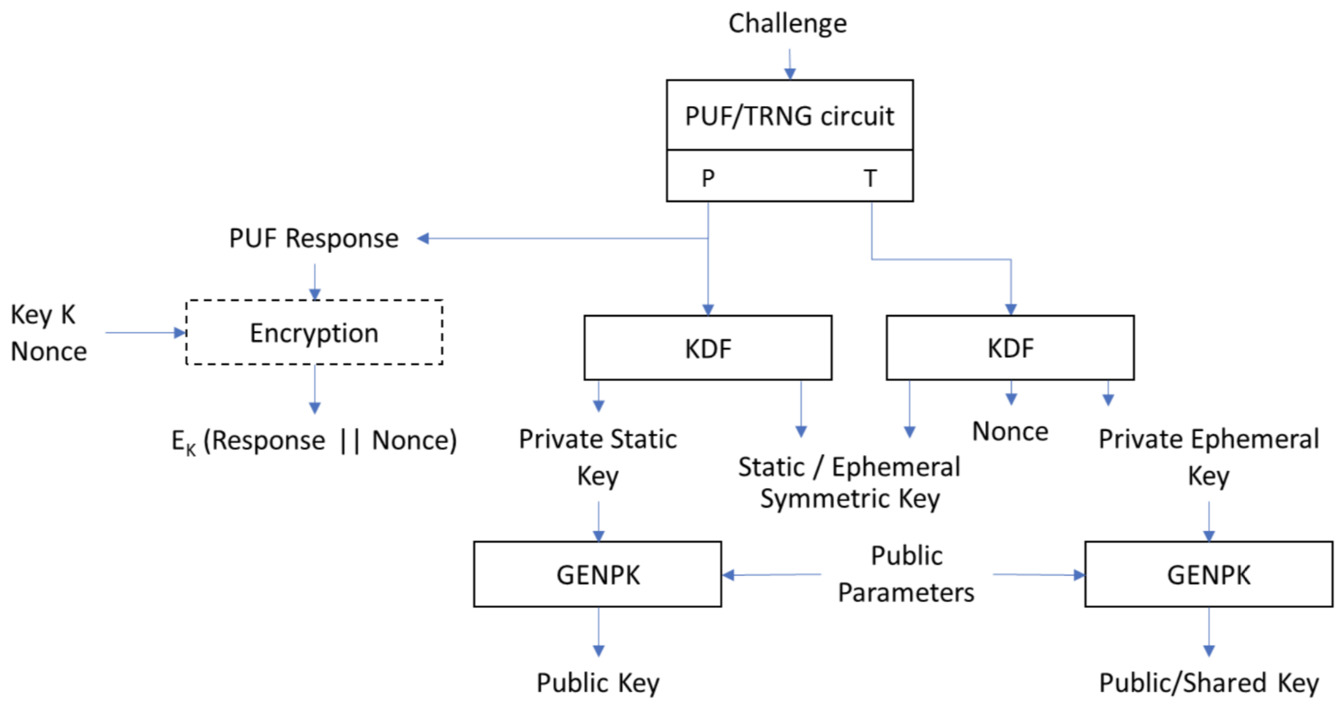
Authentication for Embedded Devices (1)

Our approach

- To simplify key management on the embedded device
- Module built on combined PUF/TRNG circuit
  - Authentication based on PUF, since it provides randomness already intrinsically present in the device and utilizes the fact the generated response is unique per device
  - Static/private keys generated using the PUF, the key that never leaves the device utilizes all the advantages of PUF
  - Ephemeral, one-time, session keys that are shared with other communicating parties generated using TRNG,

Authentication for Embedded Devices (2)

Embedded module for secure authentication and communication

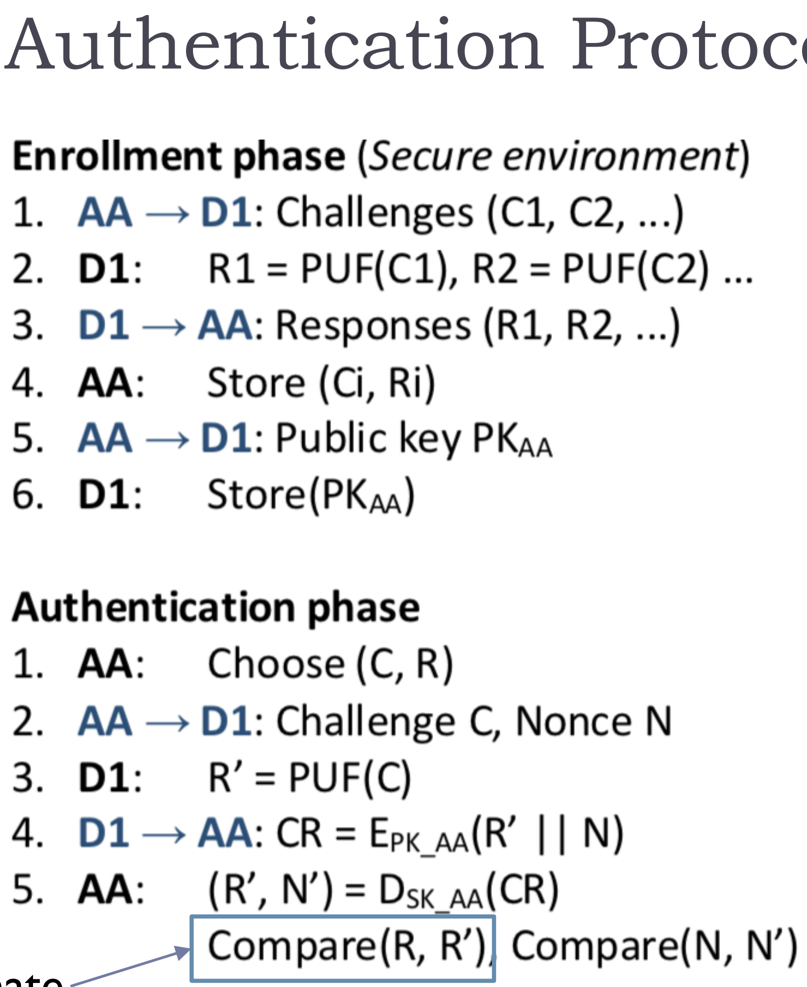


Authentication against Central Authentication Authority (1)

- Before any communication is allowed the connected device has to be properly authenticated
- PUF responses unique per each device, and intrinsically random, make the PUF the ideal cryptographic primitive for device authentication
- The authentication protocol consists of two phases
  - Secure enrollment phase
  - Authentication phase

Authentication against Central Authentication Authority (2)

- "Classical" PUF Authentication
- Response encrypted using an asymmetric cipher
  - Challenge-response pairs can be reused
  - However, needs asymmetric cipher
- Freshness of response ensured using Nonce



Authentication Protocol

Enrollment phase (*Secure environment*)

1. **AA → D1**: Challenges (C1, C2, ...)
2. **D1**: R1 = PUF(C1), R2 = PUF(C2) ...
3. **D1 → AA**: Responses (R1, R2, ...)
4. **AA**: Store (Ci, Ri)
5. **AA → D1**: Public key PK_{AA}
6. **D1**: Store(PK_{AA})

Authentication phase

1. **AA**: Choose (C, R)
2. **AA → D1**: Challenge C, Nonce N
3. **D1**: R' = PUF(C)
4. **D1 → AA**: CR = E_{PK_AA}(R' || N)
5. **AA**: (R', N') = D_{SK_AA}(CR)
   $\boxed{\text{Compare(R, R')}}$, Compare(N, N')

Approximate

Authentication against Central Authentication Authority (3)

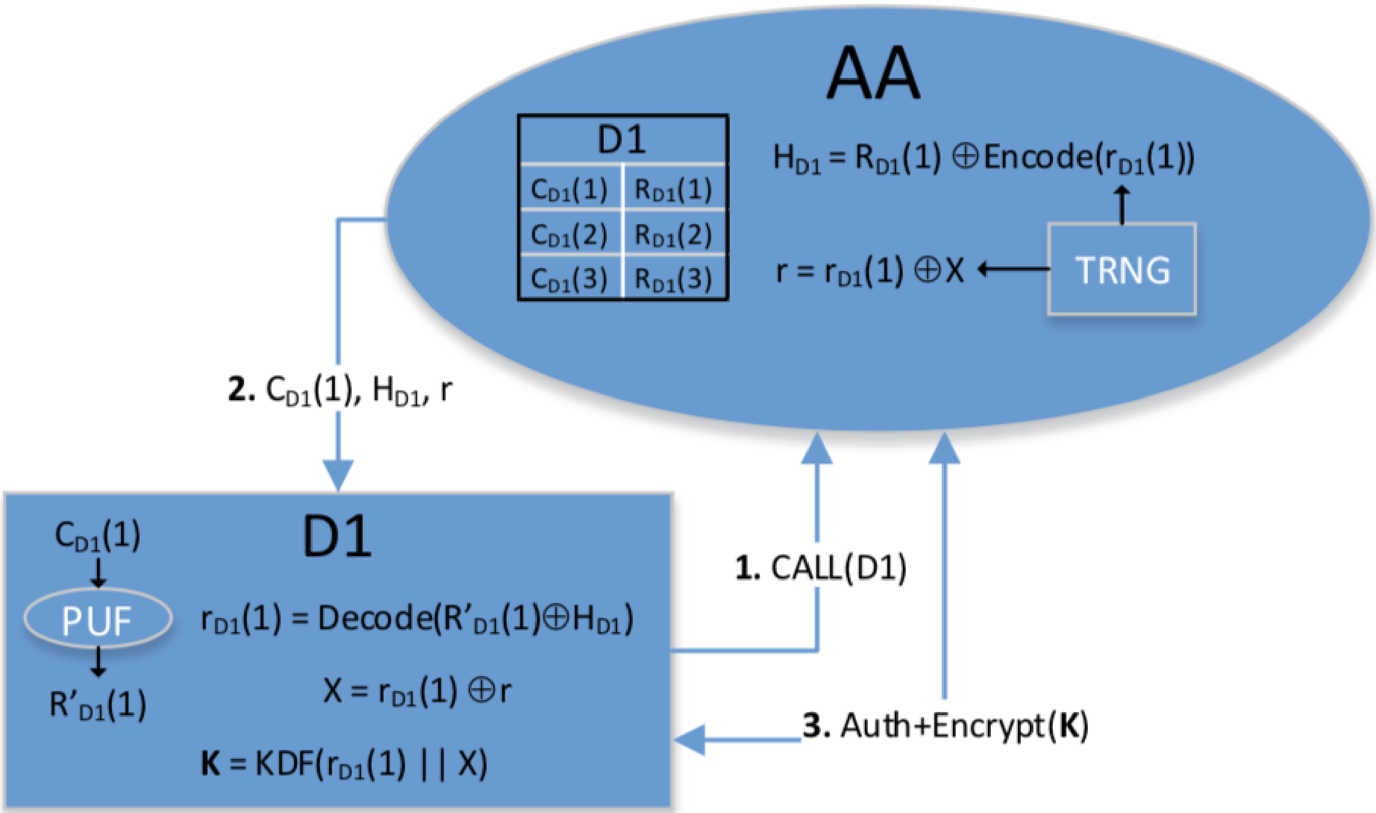
- Symmetric ciphers only
  - PUF used in key generator mode
- Helper string H_{D1} generated by AA ad hoc
- Error Correcting Code used
- Random codeword
  - $r_{D1}(1)$ = TRNG
  - Encode($r_{D1}(1)$) codeword
  - PUF response masked by random X
  - PUF can be reused

Enrollment phase (*Secure environment*)

1. **AA** $\rightarrow$ **D1**: Challenges (C1, C2, ...)
2. **D1**: R1 = PUF(C1), R2 = PUF(C2) ...
3. **D1** $\rightarrow$ **AA**: Responses (R1, R2, ...)
4. **AA**: Store (Ci, Ri)

Authentication against Central Authentication Authority (4)

Symmetric Authentication Protocol



AA
D1
$C_{D1}(1)$ | $R_{D1}(1)$
$C_{D1}(2)$ | $R_{D1}(2)$
$C_{D1}(3)$ | $R_{D1}(3)$
$H_{D1} = R_{D1}(1) \oplus \text{Encode}(r_{D1}(1))$
TRNG
$r = r_{D1}(1) \oplus X$
2. $C_{D1}(1)$, H_{D1}, r
1. CALL(D1)
3. Auth+Encrypt(**K**)
D1
$C_{D1}(1)$
PUF
$R'_{D1}(1)$
$r_{D1}(1) = \text{Decode}(R'_{D1}(1) \oplus H_{D1})$
$X = r_{D1}(1) \oplus r$
$\mathbf{K} = \text{KDF}(r_{D1}(1) \,||\, X)$

Mutual Device Authentication (1)

- AA provides Mutual Device Authentication
  - storing pre-generated challenge-response pair(s)
  - trusted 3rd party
- Establish a shared symmetric key **K**
  - conventional symmetric authenticated and encrypted session can follow afterwards → TRNG Used
  - session key derivation process can be performed using standard or light-weigh block ciphers
- Use the PUFs in both devices
  - but no PUF response transmitted over the network

Enrollment phase (*Secure environment*)

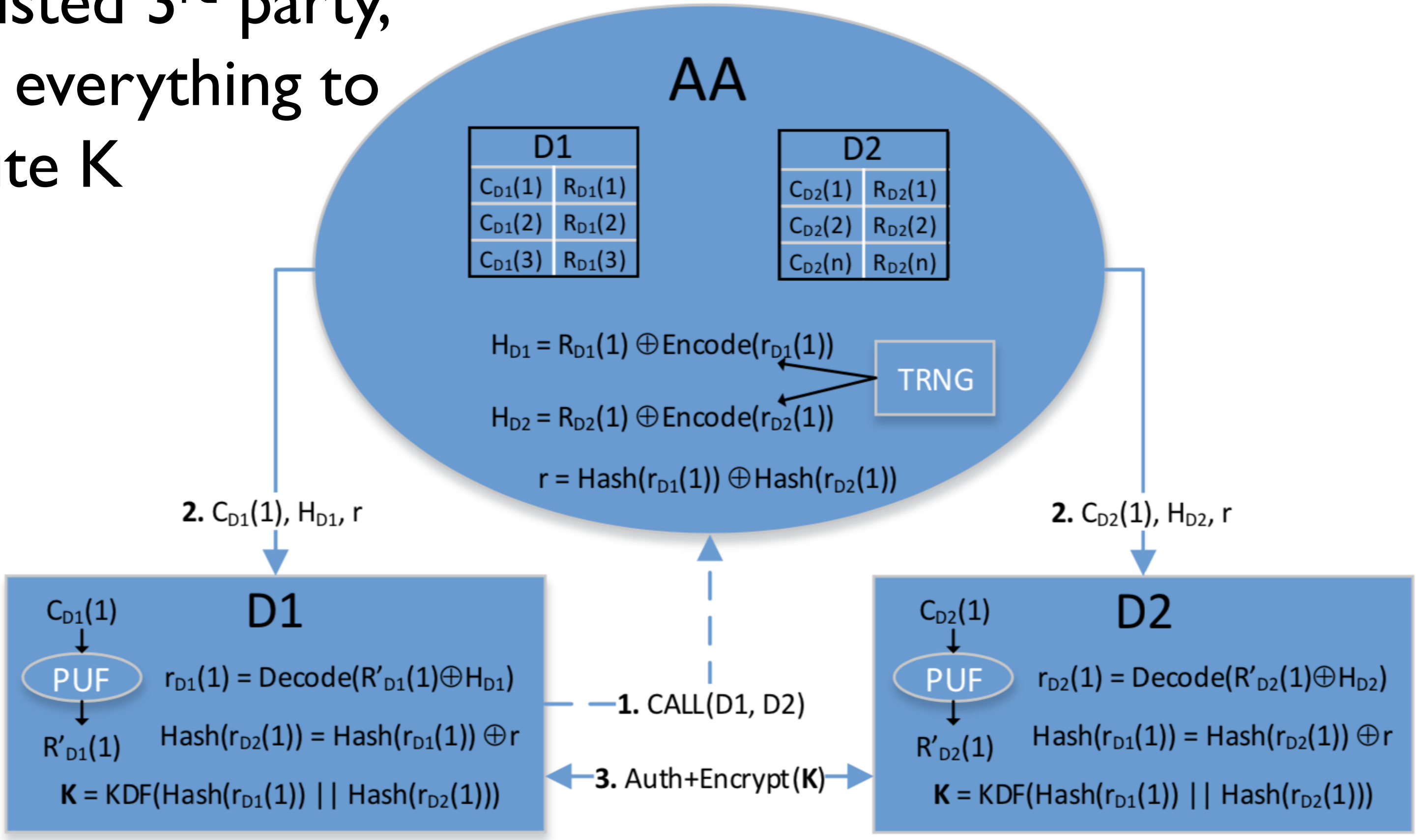
1. **AA → D1**: Challenges (C1, C2, ...)
2. **D1**: R1 = PUF(C1), R2 = PUF(C2) ...
3. **D1 → AA**: Responses (R1, R2, ...)
4. **AA**: Store (Ci, Ri)

Mutual Device Authentication (2)

&

Secure communication

- AA trusted 3rd party, knows everything to compute K



AA
D1
$C_{D1}(1)$ $R_{D1}(1)$
$C_{D1}(2)$ $R_{D1}(2)$
$C_{D1}(3)$ $R_{D1}(3)$
D2
$C_{D2}(1)$ $R_{D2}(1)$
$C_{D2}(2)$ $R_{D2}(2)$
$C_{D2}(n)$ $R_{D2}(n)$
$H_{D1} = R_{D1}(1) \oplus \mathrm{Encode}(r_{D1}(1))$
TRNG
$H_{D2} = R_{D2}(1) \oplus \mathrm{Encode}(r_{D2}(1))$
$r = \mathrm{Hash}(r_{D1}(1)) \oplus \mathrm{Hash}(r_{D2}(1))$
2. $C_{D1}(1)$, H_{D1}, r
2. $C_{D2}(1)$, H_{D2}, r
D1
$C_{D1}(1)$
PUF
$R'_{D1}(1)$
$r_{D1}(1) = \mathrm{Decode}(R'_{D1}(1) \oplus H_{D1})$
$\mathrm{Hash}(r_{D2}(1)) = \mathrm{Hash}(r_{D1}(1)) \oplus r$
$K = \mathrm{KDF}(\mathrm{Hash}(r_{D1}(1)) \,||\, \mathrm{Hash}(r_{D2}(1)))$
1. CALL(D1, D2)
3. Auth+Encrypt(K)
D2
$C_{D2}(1)$
PUF
$R'_{D2}(1)$
$r_{D2}(1) = \mathrm{Decode}(R'_{D2}(1) \oplus H_{D2})$
$\mathrm{Hash}(r_{D1}(1)) = \mathrm{Hash}(r_{D2}(1)) \oplus r$
$K = \mathrm{KDF}(\mathrm{Hash}(r_{D1}(1)) \,||\, \mathrm{Hash}(r_{D2}(1)))$

Asymmetric Schemes in Embedded Devices

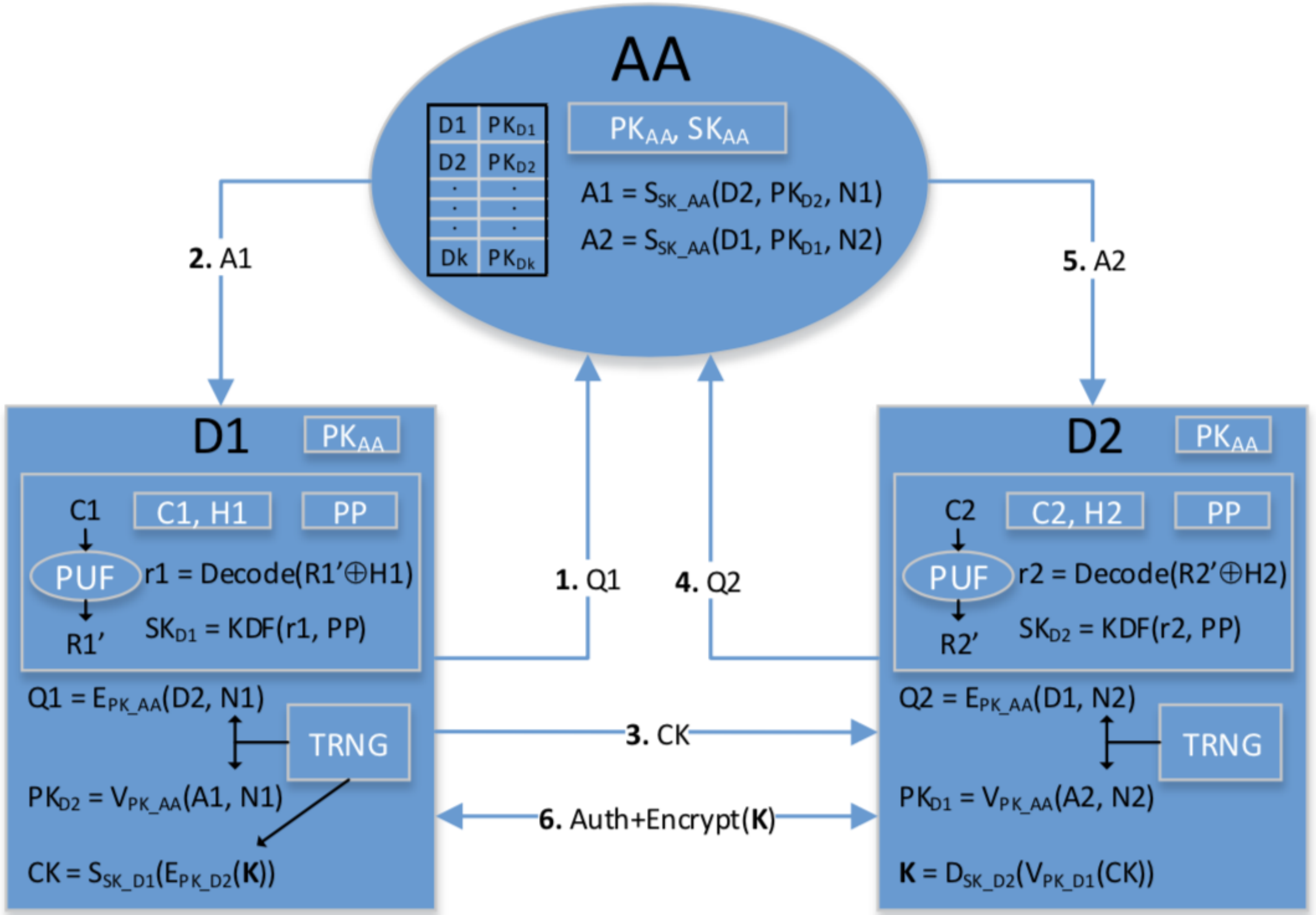
- No shared keys and the keys do not need to be transferred over secure channel
- Using PUF we do not need to store the secret key on the device, instead, the key is generated during initialization phase (e.g. on boot of the device or after longer period of inactivity)
- ElGamal encryption scheme is proposed, since there are no specific requirements on private keys (such as requirement of private numbers in RSA)
- For digital signatures, either ElGamal, or DSA or ECDSA can be used
- A private-public key pair of each device must be generated and the public key stored in the Authentication Authority

Asymmetric mutual authentication of devices (1)

Enrollment phase (*Secure environment*)

1. **AA $\longrightarrow$ D1**: Public parameters PP, Public key PK_{AA}
2. **D1**: Store PP, PK_{AA}
   Choose challenge C
   R = PUF(C)
   r = TRNG() so that Encode(r)
   is a random codeword
   H = Encode(r) $\oplus$ R
   Store (C, H)
   Private key SK_{D1} = KDF(r, PP)
   Public key PK_{D1} = GENPK(SK_{D1}, PP)
1. **D1 $\longrightarrow$ AA**: Public key PK_{D1}
2. **AA**: Store (D1, PK_{D1})

Asymmetric mutual authentication of devices (2)



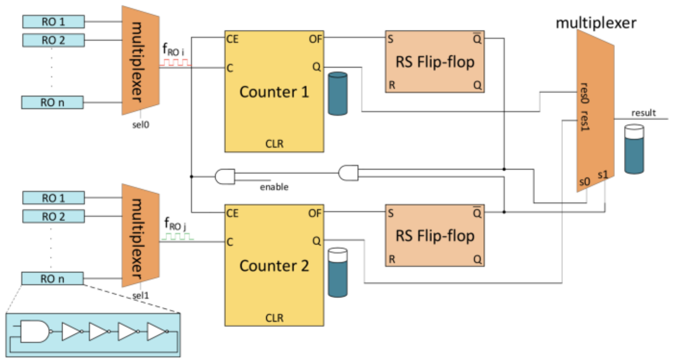
AA
D1 | PK_{D1}
D2 | PK_{D2}
Dk | PK_{Dk}
PK_{AA}, SK_{AA}
$A1 = S_{SK\_AA}(D2, PK_{D2}, N1)$
$A2 = S_{SK\_AA}(D1, PK_{D1}, N2)$
2. A1
5. A2
1. Q1
4. Q2
3. CK
6. Auth+Encrypt(**K**)
D1
PK_{AA}
C1
C1, H1
PP
PUF
$r1 = \text{Decode}(R1' \oplus H1)$
R1'
$SK_{D1} = \text{KDF}(r1, PP)$
$Q1 = E_{PK\_AA}(D2, N1)$
TRNG
$PK_{D2} = V_{PK\_AA}(A1, N1)$
$CK = S_{SK\_D1}(E_{PK\_D2}(\mathbf{K}))$
D2
PK_{AA}
C2
C2, H2
PP
PUF
$r2 = \text{Decode}(R2' \oplus H2)$
R2'
$SK_{D2} = \text{KDF}(r2, PP)$
$Q2 = E_{PK\_AA}(D1, N2)$
TRNG
$PK_{D1} = V_{PK\_AA}(A2, N2)$
$\mathbf{K} = D_{SK\_D2}(V_{PK\_D1}(CK))$

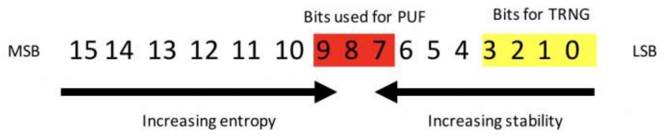
Authenticity and Integrity of the message

- MAC (Message Authentication Code) algorithm is used in cases where the authenticity and integrity of the message is critical
- Similar to asymmetric digital signatures schemes, but allows more efficient implementation using symmetric schemes
- Block ciphers can be implemented using same building blocks as discussed on previous slides
- Once the devices are mutually authenticated and have exchanged the shared key as discussed in the sections above, the communication may start, including message authentication

Case Study – ROPUF/TRNG circuit (1)

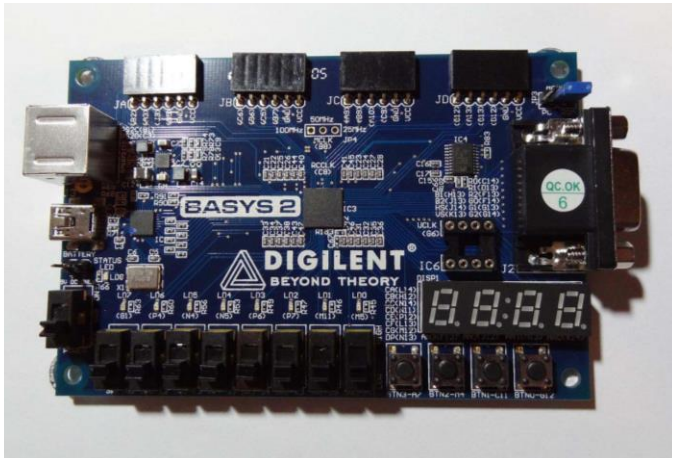
- The design provides more than one bit (typically 3 bits) of PUF response from one pair of ROs
- More bits of PUF response is needed because of the correction using ECC
- TRNG is provided at the same time
- Approx. 450 bits can be generated in one run of the ROPUF - single run of the circuit is enough for all of the proposed protocols



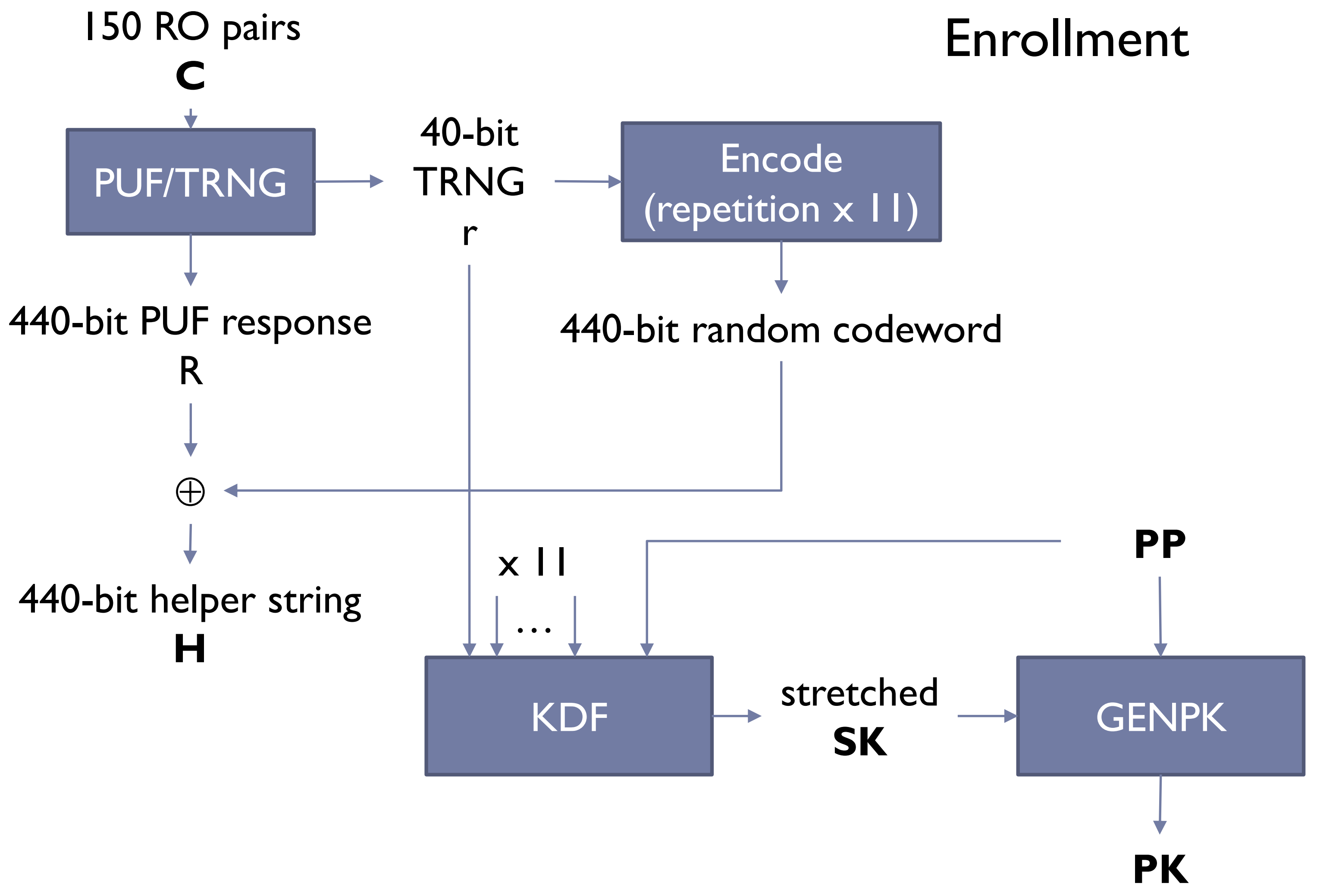


Case Study – ROPUF/TRNG circuit (2)

- Using simple repetition code with 11-bit groups, we have 40 bits – more efficient ECC can be used, or same challenge multiple times with a random codeword can be reused
- By reusing the challenge with 11 random codewords, we get up to 440 bits.
- Using a KDF, this can be stretched to longer key lengths



Case Study – Asymmetric Key Generation



Conclusion

- Combined PUF/TRNG circuit exploited
- Flexibility of use
  - Symmetric/lightweight cryptography
  - Asymmetric cryptography if needed
    - Private key not stored anywhere
- Authentication against a trusted 3rd party, the Authentication Authority
- Mutual authentication of devices in a network
- Case Study shows use of ROPUF/TRNG
  - 150 pairs of ROs, PUF=f1/f2*2^16
  - Multi-bit extraction from each pair

References

- Viktor Fischer. A closer look at security in random number generators design. In International Workshop on Constructive Side-Channel Analysis and Secure Design, pages 167–182. Springer, 2012.
- Werner Schindler. Random number generators for cryptographic applications. In Cryptographic Engineering, pages 5–23. Springer, 2009.
- Viktor Fischer and Milos Drutarovsky. True random number generator embedded in reconfigurable hardware. In International Workshop on Cryptographic Hardware and Embedded Systems, pages 415–430. Springer, 2002.
- Norbert Deak, Tamas Gyorfi, Kinga Marton, Lucia Vacariu, and Octavian Cret. Highly efficcient true random number generator in FPGA devices using phase-locked loops. In 2015 20th International Conference on Control Systems and Computer Science, pages 453–458. IEEE, 2015.
- Kohlbrenner, Paul, and Kris Gaj. "An embedded true random number generator for FPGAs." Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays. ACM, 2004.
- T. E. Tkacik, “A hardware random number generator,” in Proceedings of the Cryptographic Hardware and Embedded Systems (CHES '02), B. S. Kaliski Jr., Ç. K. Koç, and C. Paar, Eds., vol. 2523 of Lecture Notes in Computer Science, pp. 450–453, Springer, Redwood Shores, Calif, USA, 2003.
- Kirkpatrick, Michael S., Elisa Bertino, and Sam Kerr. "PUF ROKs: generating read-once keys from physically unclonable functions." Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research. ACM, 2010.
- Suh, G. Edward, Charles W. O'Donnell, and Srinivas Devadas. "Aegis: A single-chip secure processor." IEEE Design & Test of Computers 24.6 (2007).
- G Edward Suh and Srinivas Devadas. Physical unclonable functions for device authentication and secret key generation. In Proceedings of the 44th annual Design Automation Conference, pages 9–14. ACM, 2007
- Kerry A. McKay: Report on Lightweight Cryptography – NIST publication, available online https://doi.org/10.6028/NIST.IR.8114
- Kodýtek, Filip, and Róbert Lórencz. "A design of ring oscillator based PUF on FPGA." Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2015 IEEE 18th International Sympo[sium on. IEEE, 2015.
- Kodýtek, Filip, Róbert Lórencz, and Jiří Buček. "Improved ring oscillator PUF on FPGA and its properties." Microprocessors and Microsystems (2016).
- Buchovecká, Simona, Róbert Lórencz, Filip Kodýtek, and Jiří Buček. “True Random Number Generator Based on ROPUF Circuit." Digital System Design (DSD), 2016 Euromicro Conference on. IEEE, 2016.
- Buchovecká, Simona, Róbert Lórencz, Filip Kodýtek, and Jiří Buček. "True random number generator based on ring oscillator PUF circuit." Microprocessors and Microsystems 53 (2017): 33-41