IoT 機器向けの PUF ベース軽量認証プロトコルの提案

1250373　溝口 洸熙　【 セキュリティシステム研究室 】

1　はじめに

情報通信システムのセキュリティは重要であり，IoT の普及によりその重要性が増している．IoT 機器は計算能力や消費電力の制約があり，対象鍵・非対称鍵暗号方式を用いた認証技術の適用が難しい．さらに，物理攻撃やプライバシー侵害のリスクも懸念され，不揮発性メモリ (NVM) に保存されている認証情報が漏洩するリスクもある．これらのリスクを解消する技術として PUF がある．PUF は，物理特性を利用した関数である．入力 C とその出力 R の組み CRP には，PUF による一意性，再現性，複製困難性，一方向性，タンパリング耐性があるため，PUF は，物体固有の暗号学的ハッシュ関数のように振る舞う．本研究では，IoT 環境における課題を解決した軽量な CSS(Client-Server-System) 型の PUF ベース認証プロトコルを提案する．

2　関連研究

従来の PUF を用いた認証・鍵交換方式として，2023 年に提案された秘密分散を用いた認証方式を挙げる [1]．この方式は，無加工 CRP の送受信や Verifier 側への保存を必要としないため，機械学習攻撃等から CRP の機密性を保護することができる．しかし，Prover 側は，認証毎に変わる，ID の匿名化のための内部情報を，保持する必要がある．

3　提案方式

$\texttt{PUF}(x)$ を PUF 関数，$H(x)$ をハッシュ関数とする．

初回登録段階は，セキュアな通信路を介して Verifier と Prover が認証情報を共有する．Verifier は 3 つのチャレンジ c_1，c_2，c_3 を乱数から生成し，Prover へ送信する．Prover は，c_1，c_2，c_3 を PUF 関数に入力し，それぞれの出力 r_1，r_2，r_3 から，$r_1 \oplus r_2$，$r_2 \oplus r_3$ を生成する．その後，生成した値を Prover の ID と一緒に Verifier へ送信する．Verifier は，Prover から受信した ID と $r_1 \oplus r_2$，$r_2 \oplus r_3$ を保存する．

認証段階は，Prover が一時識別子 $\texttt{ID}_t$ を

$$\texttt{ID}_t = H\left(\texttt{ID} \,||\, t\right) \quad ,\text{where } t = \left\lfloor \frac{\text{UNIX TIME}}{x} \right\rfloor$$

で生成し，Verifier へ送信する．ここで x は更新頻度を表す定数（秒）である．Verifier 側は，データベースに保存された全ての Prover の ID から，UNIX TIME に合わせて $\texttt{ID}_t$ を生成し，Prover から受信した $\texttt{ID}_t$ と比較する．一致した Prover の ID に対して，c_1，c_2，c_3，$r_1 \oplus r_2$，$r_2 \oplus r_3$ を取得し，$H(r_1 \oplus r_2)$ と，新しく乱数から生成したチャレンジ c_4 を含めたチャレンジ $c_n, (n = 1, 2, 3, 4)$ を送信する．Prover は，c_n を PUF 関数に入力し，それぞれの出力 r_n を得る．その後，$H(r_1 \oplus r_2)$ を生成し，Verifier から受信した $H(r_1 \oplus r_2)$ と比較する．この一致を持って Prover は Verifier を認証する．一致した場合，Prover は $H(r_2 \oplus r_3)$，$r_2 \oplus r_4$ を生成し，Verifier へ送信する．Verifier は，Prover から受信した $H(r_2 \oplus r_3)$ と，保存している $r_2 \oplus r_3$ をハッシュ化した $H(r_2 \oplus r_3)$ を比較し，一致した場合 Prover を認証する．認証が成功した場合，Verifier と Prover はセッション鍵を生成する．Prover は，生成した r_1，r_4 から $H(r_1 \oplus r_4)$ を生する．Verifier は $(r_2 \oplus r_3) \oplus (r_2 \oplus r_4)$ を計算し，その結果から得られた $r_3 \oplus r_4$ と保存している $r_1 \oplus r_2$，$r_2 \oplus r_3$ を全て XOR 演算し，ハッシュ化することで $H(r_1 \oplus r_4)$ のセッション鍵を生成する．

その後，Verifier は，データベースのデータを $r_1 \oplus r_2$ から $r_2 \oplus r_3$ へ，$r_2 \oplus r_3$ から $r_3 \oplus r_4$ へ上書きし，認証を終了する．

4　評価・考察

安全性を評価した結果，DoS 攻撃，中間者攻撃，リプレイ攻撃，ID 平文通信によるプライバシー侵害，機械学習攻撃等の脅威に対して耐性があり，前方秘匿性についても保証できることが分かった．

性能評価として，従来方式 [1] と提案方式の，Prover 側におけるハッシュ関数適用回数，擬似乱数生成回数，認証 1 回あたりの通信量と実行時間を比較した．

表 1　Prover 側における性能評価

方式	$H(\cdot)$	rand	CM(B)	RT(ns)
[1]	10	2	≈ 464	2,552
提案方式	4	0	≈ 176	1,084

$H(\cdot)$：ハッシュ関数適用回数，rand：擬似乱数生成回数，CM：認証 1 回あたりの通信量 (単位: Byte)，RT：認証 1 回あたりの実行時間 (単位: ns)．PUF については模擬的な PUF 関数を用いて評価し，送受信時のオーバヘッドを考慮していない．

結果より，提案方式は，従来方式に比べて，計算量を抑えて，通信量が少なく，短い実行時間で認証と鍵交換を実現できることがわかる．

今後の課題として，Verifier 側の定数時間毎のデータ更新にかかる負荷を軽減するための最適化等が挙げられる．

参考文献

[1] K Nimmy et al. “A novel lightweight PUF based authentication protocol for IoT without explicit CRPs in verifier database”. In: *Journal of Ambient Intelligence and Humanized Computing* 14.5 (2023), pp. 6227–6242.