**1. Related Work: Overview of Diffie-Hellman and RSA Technologies**

In modern cybersecurity, key management is the foundation of secure communication. The Diffie–Hellman (D–H) key exchange protocol and the RSA public-key encryption algorithm are two fundamental pillars of modern cryptography, offering essential solutions to the problems of key distribution and digital identity verification.

**(1) Diffie–Hellman Key Exchange Protocol**

The Diffie–Hellman (D–H) protocol, proposed in 1976, enables two communicating parties to establish a shared secret key over an insecure channel without directly transmitting it [1]. The security of the protocol relies on the computational hardness of the discrete logarithm problem, meaning that deriving private keys from public information is computationally infeasible [2].  
In practice, both parties generate private keys and corresponding public keys based on a shared modulus and generator. After exchanging public keys, each side can independently compute the same shared secret key.  
However, the D–H protocol does not inherently provide authentication, making it vulnerable to Man-in-the-Middle (MITM) attacks [3]. As a result, it is commonly combined with digital signatures or certificates in protocols such as TLS to ensure both confidentiality and authenticity.

**(2) RSA Public-Key Cryptosystem**

The RSA algorithm, introduced by Rivest, Shamir, and Adleman in 1978, was the first practical public-key cryptosystem that supports both encryption and digital signatures [4]. RSA’s security is based on the mathematical difficulty of prime factorization of large integers—specifically, decomposing a modulus into its two large prime factors [5].  
RSA uses a pair of keys: the public key is used for encryption or signature verification, while the private key is used for decryption or signing. Because the private key cannot be feasibly derived from the public key, RSA provides robust asymmetric encryption and authentication.  
Due to its versatility and proven reliability, RSA is widely deployed in systems such as PGP for secure email, TLS/SSL for web protocols, SSH, digital certificates, and two-factor authentication systems like RSA SecurID [5]. Despite the emergence of newer algorithms, RSA remains one of the most trusted cryptographic standards in modern cybersecurity infrastructure.

**2. Methods: Analysis of Attacks Against D–H and RSA**

To better understand the security boundaries of these algorithms, we examine two representative and implementable attack techniques:

* the Man-in-the-Middle (MITM) attack against the D–H key exchange protocol, and
* the factorization attack on RSA using Pollard’s rho algorithm.

These examples demonstrate how weaknesses in protocol design and computational limits can be exploited, as well as the countermeasures that can mitigate such risks.

**2.1 Man-in-the-Middle Attack on Diffie–Hellman Protocol**

**2.1.1 Attack Principle**

The D–H protocol performs key agreement without authenticating the communicating parties, leaving it inherently vulnerable to MITM attacks [3]. An attacker can intercept public key exchanges and impersonate each party to the other, establishing independent shared keys with both sides. This allows the attacker to decrypt, modify, and re-encrypt communications invisibly.

**2.1.2 Attack Procedure**

The MITM attack typically follows these steps [3]:

1. Interception and Impersonation:  
   When Alice and Bob exchange their D–H public keys, the attacker (Mallory) intercepts both messages.
2. Session Substitution:  
   Mallory generates her own private and public keys. She sends her public key to Alice pretending to be Bob, and to Bob pretending to be Alice.
3. Key Derivation:
   * Alice computes a shared key with Mallory’s public key, assuming it came from Bob.
   * Bob does the same, computing a shared key with Mallory.
   * Mallory now holds two distinct keys—one shared with Alice and another with Bob.
4. Traffic Manipulation:  
   Mallory can decrypt any message sent by Alice, read or alter its content, and re-encrypt it using the second shared key before forwarding it to Bob (and vice versa). Both legitimate users remain unaware of the breach.

**2.1.3 Implementation and Defense**

Implementing this attack requires intercepting and modifying network traffic, often through ARP spoofing or DNS hijacking. The core vulnerability lies in the absence of authentication in the base D–H protocol.

Defensive strategies include:

* Digital Certificates (PKI):  
  Using certificates signed by a trusted Certificate Authority (CA) to bind public keys to verified identities, as implemented in TLS/SSL.
* Static D–H Keys with Out-of-Band Verification:  
  Parties maintain fixed key pairs and verify fingerprints through secure alternative channels.

By introducing authentication, the MITM attack can be effectively mitigated without undermining D–H’s core strength in key establishment.

**2.2 Factorization Attack on RSA Using Pollard’s Rho Algorithm**

**2.2.1 Attack Principle**

RSA security depends entirely on the infeasibility of factoring a large composite number . If an attacker successfully factors , they can derive the private key from the public parameters , completely breaking the system [4].  
Among various factorization methods, Pollard’s rho algorithm is an efficient probabilistic algorithm that performs well for medium to large integers. It leverages the birthday paradox to detect a nontrivial factor of by finding collisions in a pseudorandom sequence [6].

**2.2.2 Attack Procedure**

Pollard’s rho algorithm proceeds as follows:

1. Initialization:  
   Choose initial values , and set .
2. Iterative Computation:  
   Use a pseudo-random function such as :
   * (moves twice as fast as )  
     After each iteration, compute .
3. Result Checking:
   * If , continue the process.
   * If , is a nontrivial factor of (i.e., or ), and the factorization succeeds.
   * If , restart with different initial parameters.

This cycle-detection approach is efficient because it exploits Floyd’s cycle-finding algorithm, which allows fast collision detection with minimal memory.

**2.2.3 Implementation and Defense**

Pollard’s rho algorithm’s efficiency depends on detecting cycles rapidly within the modular arithmetic space. Research shows that its running time correlates with the number of iterations and grows moderately with the size of [6].

Preventive Measures:

* Generate RSA keys with large and random primes and .
* Ensure the modulus length bits, as smaller key sizes are susceptible to factorization-based attacks.
* Use well-verified implementations to avoid side-channel or mathematical flaws.

When implemented correctly with sufficiently large key lengths, RSA remains secure against Pollard’s rho and most known attacks [7].

**References**

[1] W. Diffie and M. Hellman, “New directions in cryptography,” *IEEE Transactions on Information Theory*, vol. IT-22, no. 6, pp. 644–654, 1976.  
[2] U. M. Maurer and S. Wolf, “The Diffie–Hellman protocol,” *Designs, Codes and Cryptography*, vol. 19, pp. 147–171, Springer, 2000.  
[3] K. M. Jain, M. V. Jain, and J. L. Borade, “A survey on Man in the Middle Attack,” *International Journal of Science and Research (IJSR)*, vol. 5, no. 2, pp. 2319–7064, 2016.  
[4] X. Zhou and X. Tang, “Research and implementation of RSA algorithm for encryption and decryption,” *Proc. 6th International Forum on Strategic Technology (IFOST)*, IEEE, 2011.  
[5] S. Nisha and M. Farik, “RSA Public Key Cryptography Algorithm: A Review,” *International Journal of Scientific and Research Publications*, vol. 7, no. 2, pp. 397–402, 2017.  
[6] M. A. Budiman and D. Rachmawati, “On factoring RSA modulus using random-restart hill-climbing algorithm and Pollard’s rho algorithm,” *Journal of Physics: Conference Series*, IOP Publishing, vol. 893, pp. 1–6, 2017.  
[7] M. Mumtaz and L. Ping, “Forty years of attacks on the RSA cryptosystem: A brief survey,” *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 22, no. 4, pp. 603–626, 2019.