A Literature Review of Security and Cryptographic Protocols

Nancy N. Mogire

Portfolio Preparation

Department of Information and Computer Sciences

University of Hawaii at Manoa

Honolulu, USA

nmogire@hawaii.edu

Abstract

The study of security protocols is the foundation for designing in-built security in communication systems. In this literature survey we seek to understand current frameworks that guide security protocol design and analysis and also the changing security needs in computing systems as they evolve into elaborate cyberphysical networks. Our goal is to derive a formal framework for design, analysis and verification of cyberphysical protocols. The problem investigated in the research work that follows after this literature review is that current formal frameworks for modelling security protocols are not made for cyberphysical networks.

Index Terms

Cyberphysical Network; Ceremonies; Social Computation; Actor Networks; Protocol Verification; Protocol State Machine; Cryptographic Protocols; Protocol Proving Axioms; Protocol Process; Adversary; Intruder; Protocol Taxonomy; Protocol Correctness; Key Exchange; Key Establishment; Formal Methods; Security properties; Protocol participants; Session Key; Session;

I. Introduction and background

Over the recent years, researchers have done work on protocol design and analysis but often modelling frameworks were focused on cyber-based protocols since the computation in cyberspace was somewhat clearly separated from that in physical space. However in recent years we see computing systems designed to interact with people and physical objects in a more augmented way. Ellison[9] in a 2007 Microsoft

Research white paper terms this interaction as a 'ceremony'. He describes the disadvantage of frameworks made for computer-to-computer protocols when used to analyze ceremonies as the fact that it results in too many assumptions. This is because in computer-to-computer protocols several actions are considered out-of-band for the protocol during analysis. One example he describes is the HTTPS protocol which requires provisioning of the client computer with the PKI root key. This activity is not typically analyzed as part of the protocol but rather it is assumed that this has already happened securely and is out-of-band for the protocol's analysis. Essentially a lot of interaction in physical space is not analyzed. This is a disadvantage because in the increasingly cyberphysical world computation is really in physical space as much as it is in cyberspace.

This realization motivated the concept of actor networks where each person or object in the network is regarded importantly as an actor. Latour[16] in his work on actor networks discusses that objects whether artificial or natural have agency since they all react in various ways. He opines that 'social' is not merely the characteristic of some movement but rather;

"social, for ANT, is the name of a type of momentary association which is characterized by the way it gathers together into new shapes"

It turns out that as early as the late 1970s Carl Hewitt[14] was also already thinking about objects as actors within their work on artificial intelligence albeit in a slightly different way. In their work actors referred to processors. Meadows and Pavlovic[21] began to analyze the idea of actors in the sense that all protocol participants are actors and should be seen as such for protocol analysis. In their work as well 'actors' refers to any participant including both those that process and those that simply react. Each actor's actions, cause changes in state of the network for other actors, conclusions that each actor can make and how the overall security goal is achieved. Additionally, in physical space a lot of properties change in the short term. People and things move, new devices enter the vicinity, some devices react to human action such as motion and speech and many other dynamics. Hence to analyze protocols for actor networks we need a framework developed with the dynamics of cyberphysical space in mind to achieve the necessary precision and robustness.

A. Achievements and outline of the report

This report achieves the goal of providing a brief overview of the science of cryptographic and security protocol design and also builds up to the starting point of our subsequent research in cyberphysical

protocol modelling. The report is organized as follows. In Section 2 we begin with an introduction to the main concepts behind cryptographic protocols in part A. In part B we discuss the role of cryptography in protocols design. Next in part C we take a historical look at the use of protocols, discussing some classical examples. In the next part D we now conclude the section with a look at protocol failure and attacks on protocols.

In Section 3 we focus on protocol design and analysis. In part A we discuss various approaches both formal and semi-formal used in protocol description and analysis. In Section IV we provide an introductory overview of cyberphysical networks and the changing security expectations from communication protocols. We overview the security need in cyberphysical networks and discuss some of the works that motivated the concept of actor networks. This leads us to the point where our research is anchored. In section V we now discuss some formalisms and frameworks directly or closely related to our work including the protocol derivation logic framework, hoare logic as a process algebra and then actor networks formalism. Overall we attempt to draw a picture of why it is necessary to design a language of protocol design, description and analysis particularly for cyberphysical space and the approach and theoretical background we shall use in achieving this goal.

II. Introduction to Cryptographic Protocols

A. Overview of Protocol Concepts

A protocol is defined as ”a system of rules that explain the correct conduct and procedures to be followed in formal situations” (Merriam Webster Online, 2015). Within the context of computing, there are several types of protocols depending on the network layer at which each one occurs. Two examples are hypertext transfer protocol(HTTP) and simple mail transfer protocol(SMTP). Each protocol has its own formal definition. Every protocol is defined within the context of some communication architecture. In this document when discussing cyber based protocol we focus on a generic scenario where two client computers we call A and B, want to be able to exchange messages securely over a cyberchannel. Since messages are transported in packets across various network components and locations, they are susceptible to intrusion. Intruders can reprogram network components at convenient points for them to intercept messages and either prevent them from reaching the intended receiver or alter them and relay an altered version. Hence the cyberchannel is said to be insecure.

The problem A and B face is the need to establish a shared session key to enable them exchange encrypted messages and hence secure their communication over the network during the session. The two principals A and B can either make use of a trusted server to randomly generate and distribute the session key to them or they could use public key cryptography to each privately generate the shared key. The purpose of separating communication into sessions is to restrict access to each message run to a specific set of principals and to a reasonable timeframe. It is assumed that an attacker eventually learns the key given a sufficient amount of time. The important feature of this session key is that it should be unpredictable to the adversary.

In protocols security, we assume perfect cryptography and abstract away from the details of it. Instead we focus on the primary task of a protocol which is to make the key K^{AB} known to A and B and no other party except possibly the trusted server S. The idea then is to authenticate A and B so they receive information about the key. Authentication problem arises from the fact that the adversary controls the communication channel such that they are able to alter, insert, delete or re-route messages in the protocol. The extent of such ability varies based on various properties of networks and as such the specific impact is not always obvious.

Beyond making the key available to the principals there are certain other properties that a protocol should guarantee in order for such key to be expected to provide security to the session. For instance the protocol should assure both A and B that K^{AB} is newly generated, which is known as the freshness property and it ensures that a principal does not receive a replay of an old key that an adversary may already know. Protocols should also assure confidentiality and integrity. Every protocol is designed with some specific set of primary goals based on what type of security is desired for the specific communication. These goals are given a more detailed treatment later in this section. As will become apparent, the goals of a protocol determine whether the protocol is considered secure or not based on whether these goals are achieved.

So far, the tradition in protocol design is to specify the structure of a successful protocol run i.e. the expected message sequence. As such, no error handling is envisioned and protocols do not provide for message validation. Additionally, since the actions of principals are not detailed in the protocol it is not obvious how principals deal with decision making in case several options come up e.g. of different message constructions. However, there is an established practise around protocol analysis that is intended to prove protocols secure. While there has been work on writing protocol proofs, a big part of protocol

analysis is attempting to design attacks on protocols and for as long as an attack is not found on a given protocol it is assumed to be secure.

Protocols can be classified in various ways but three major criteria are; whether or not shared keys are already established when the protocol begins; How the session key is generated and how many users the protocol is designed to serve. With regard to the first criterion, there are three cases. The first is where protocols are designed for the scenario where the principals who want to generate a session key already share a secret key. The second case is where the principals have certified public keys. The last case is where an online server who is a trusted third party mediates the key exchange [2].

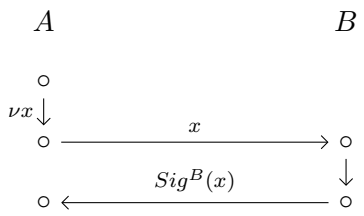
The second mode of classification also has three types of protocols. The first is key transport protocols where one of the principals generates the key to be transported to all other users of the protocol. The second type is key agreement protocols where the session key is a function of inputs by all protocol users. The third type of protocols are hybrids being key transport for some users and key agreement for others. The idea is that the session key is a function of inputs from more than one but not all users in the protocol. The subset of users that participates in generating a key are the ones referred to as principals. With respect to classification by number of users, protocols can be point to point serving only two users in any given run or they can be conference key protocols that will have multiple participants.

B. Protocol Goals

There are a number of goals that a protocol can be designed towards in any communication architecture. Here we review some of the most important goals. Boyd et al[1] categorize these goals into user-oriented goals, Key-oriented goals and enhanced goals comprising of goals that serve the function of strengthening the primary goals. Here we discuss the two primary goals.

The first goal is *key establishment*. This is where the protocol is intended to establish a fresh session key known only to the participants in the session. Menezes et al[18] define key establishment as the process by which a shared key becomes available to two or more parties for subsequent cryptographic use. Successful key establishment leads to a good session key. Boyd et al[1] describe the properties of a good session key. To begin with, from principal A's perspective a shared session key is good to use with B if A is assured of the Key's freshness and the key is only known to A, B and any trusted third parties. B determines if the session key is good by similar reasoning. A public session key is good if freshness is guaranteed and the private key only known to the owner. A key is good if it guarantees integrity i.e.

all inputs for its generation come from legitimate participants and cannot be modified by the adversary. Another goal of protocols is *entity authentication*. Menezes et al[18] provide a definition for entity authentication as follows; "For one party to acquire corroborative evidence of identity of another party involved in the protocol and that such party has actually participated". Authentication can be achieved in various ways including requiring the party to be authenticated to provide a signature on some message as in the example below:



Boyd' et al[1] break down entity authentication more specifically; a principal A is said to have knowledge of B as their peer in a protocol if A is aware of B as their claimed peer entity in the protocol. Strong entity authentication of A to B can be claimed for a protocol if B has fresh assurance that A has knowledge of B as its peer and A is operative. Mutual authentication is achieved if both entities authenticate each other in the same protocol, otherwise authentication is unilateral. Perhaps the reason Boyd et al categorize the remainder of the protocols as enhanced is because they arise in building up towards either one of the two first goals. These other goals include Key confirmation, key freshness and mutual understanding of shared key i.e. both A and B have the same key k and each is certain that the other knows the key k.

1) Authentication and Key Establishment: The problem of authentication is about finding a digital replacement of paper instruments of agreement. The digital alternatives include digital signatures used to prove authenticity of documents. Another common scenario where authentication is necessary is where a principal A wants to verify the identity of a principal B before sharing a session key for communication. Also A wants to have something in the communication that later proves to him that the responses he receives from B are fresh and not a replay from old conversations. Authentication is often achieved via the mechanism known as challenge-response. In challenge-response process a user is challenged to prove his identity by demonstrating ability to provide the correct response to a challenge and often also encrypt or decrypt with a secret key known to be in possession only of the intended parties in an exchange.
A challenge can be a timestamp, counter or nonce. It should be new for each run of the protocol.Boyd and

mathuria[2] discuss that various difficulties may be experienced depending on the type of challenge. For instance the use of timestamps as authentication challenges requires good clock synchronization which may be difficult to achieve depending on the network scenario. Counters may require a synchronized count of how many times A and B authenticate each other. The problem arises when both A and B want to initiate a protocol exection at the same time. This scenario may result in a loss of synchronization and subsequent conflicts. Also because the counter numbers continue to grow and are very large numbers, counter management may be problematic in the long run. Hence the reason why use of nonces is favored in most protocols, nonces are independent of system factors and present few management problems with the only cost being an extra message in the protocol. Attacks on cryptographic authentication can be many. Some attacks happen at the level of the protocol's logic while others may attack the cryptographic algorithm.

Diffie, Oorschot and wiener[7] introduce us to Authenticated key exchanges whose goal is to provide the protocol participants with a guarantee on each others' identities and avail the secret key to these parties and only them. In their discussion they focus on two-party authentication and assume perfect cryptography as is the norm in protocols discussion. In a successful protocol run as the authors present, protocols participants A and B exchange messages and at the end they have assurance of each other's identities and and optionally they exchange and share a secret key with each party accepting the other's identity and key. They note that a successful protocol run is not synonymous to a secure run. A successful protocol run should also have matching records of conversation at the end of the run.This implies for example that when one party's record shows a message as incoming, the other party should have the same message recorded as outgoing. Only messages relevant to authentication need match.

In addition, messages need not be recorded in the same order by both participants but they should be categorizable into sets of matching messages whereby messages from one participants are in the same sequence on both records.

C. Role of Cryptography in Protocols

From the preceeding discussion it is clear that confidentiality and integrity are the key problems in the design of protocols. Cryptography is a natural solution to these problems since it provides mathematical systems for solving privacy and authentication problems. Diffie and Hellman[19] defined privacy as the prevention of extraction of information from a communication channel by unauthorized parties and authentication as the prevention of injection of messages into a public channel by unauthorized persons.

A cryptosystem is defined as a system of invertible functions $\{S_k|k \in |K|\}$ such that S_k is a function $S_k : |p| \rightarrow |C|$ where P is the message space and $|C|$ is the ciphertext space. The goal of a cryptosystem is to make cryptanalysis of ciphertext too complex to be economical for an intruder while encryption and decryption are simple for the intended participants of the communication. The difficulty of cryptanalysis is the security of a cryptosystem. Security is computational if the cryptosystem is secure due to the cost of cryptanalysis such as if too much computing memory or runtime is required than is feasible[19]. Unconditional security is achieved when a cryptosystem is secure even with unlimited computing capacity. An example of such a system is the one-time pad where the plaintext is combined with the randomly chosen key of equal length that is used only once. This system is impractical for most scenarios due to the bulk of keys that would be needed.

Needham and Schroeder [24] discuss the use of encryption to achieve decentralized authentication by enabling principals to verify the identities of one another. Their paper outlines the goals of encryption in protocols as achieving authenticated one-way communication, authenticated interactive authentication and signed communication which guarantees integrity of content.

For their discussion, Needham and Schoeder assume perfect cryptography i.e. feasible encryption and decryption and secure keys. They also assume that the adversary controls the channel of communication. For instance the adversary can copy, replay as well as emit false messages. It emerges later that these assumptions are seen often in similar cryptographic protocol studies. The basis of authentication is the secret key of each principal. For instance an authentication server can compute identifying information from a principal's secret key. Encryption takes place before the network interface for the sender while decryption is after the the message passes the network inteface for the receiver. Different protocols use different cryptosystems some making use of public key systems and some using non-public key systems. For non-public key systems, each principal shares a secret key with the server. If principals $A\&B$ want to communicate, the initiator A generates and sends a message to the server S encrypted with their shared key, requesting a session key to use with B. The server upon verifying the identity of A computes a session key CK to be used by A and B and encrypts a copy for A and a copy for B using their respective secret keys. Then A can now send a message such that it is only comprehensible to B and B can ascertain that it originated from A. In this way a secure channel is created noting that a trusted third party i.e. the server knows the session key CK. Another note is that it is not necesary that both clients query the same server. Multiple authentication servers can be used and this should have no effect on the

communication between clients.

Diffie and Hellman contributed the idea of public key cryptosystems(PKCs). In their earliest paper on cryptography[19] they identified the necessity for cryptosystems that minimize the need a secure channel for key distribution and provide an equivalent of a digital signature. They envisioned that privacy and authentication could be achieved via a public channel which they defined as a channel whose security is inadequate to its users.

In their idea which draws from the work of Ralph Merkle, a PKC is a pair $\{E_k|k \in |K|\}$ and $\{D_k|k \in |K|\}$ such that $E_k : |M| \to |M|$ and $D_k : |M| \to |M|$ where M is a finite message space and for every $k \in K$, D_k is the inverse of E_k; for every $m \in M$, E_k and D_k are easy to compute; it is feasible to compute inverse pairs E_k and D_k from K for every $k \in K$; and finally it is hard to derive D_k from E_k for almost every $k \in K$ hence E_k can be made public without compromising D_k. The family of transformations is public information just like a resettable combination lock whose structure is public but the combination is secret. In public key cryptography, each user generates both D_k and E_k from some $k \in K$ known only to themselves, and then keeps D_k secret while placing E_k in a public directory. In this way anyone can encrypt messages for the user while only he can decrypt them.

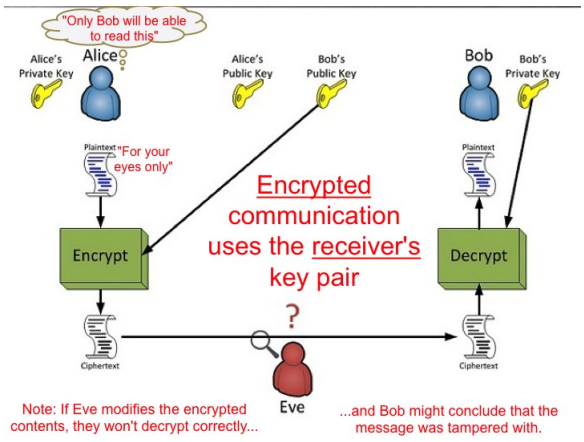


Fig. 1: public key cryptography [diagram from http://www.usna.edu/CS/si110arch/lec/l28/lec.html]

Diffie and Hellman then proposed a signature scheme for one-way authentication extending from public key cryptosystems. The defining property in public key authentication is that it must be easy for anyone to recognize the signature as authentic, but only the legitimate signer can originate an authentic signature. One-way authentication relies on some one-way function f such that for any x in the domain, it is easy

to compute $f(x)$ but for all y in the co-domain it is computationally infeasible to solve for $y = f(x)$ for any x. An example application is in password-based protocols. For example when a user creates a password pwd to log into some system, the system computes and stores a function of the password $f(pwd)$ where the function f is publicly known. Next time the users enters pwd, the system once again computes $f(pwd)$ and checks against what is stored in the password database. If an adversary obtains the database containing $f(pwd)$ he cannot obtain from it the original value pwd. Hence he cannot log into such user's account since the system does not accept $f(pwd)$ as a password.

D. Protocol Failure

There are various factors that will cause protocols to fail in security. Bird et al [20] outline some factors that make it difficult to guarantee protocol security such as protocols needing synchronization of local clocks, export restrictions on cryptography which leads to some cryptography implementations being weak and non-useability of some protocols in lower layers of the network due to size and complexity of messages. Some other problems include passwords and biometric data being transmitted in the clear and easy to guess passwords as well as phishing attacks enabled by the fact that the server does not typically authenticate itself to the user.

Dolev and Yao[22] note that while public key systems are secure against passive attackers, an improperly designed protocol could be vulnerable to an active saboteur. Simmons[28] presents a more surprising perspective that while it is true a protocol will be broken if the underlying cryptoalgorithm is broken, it is also true that a badly designed protocol can be subverted without tampering with the underlying cryptoalgorithm. Indeed his argument is that protocol failures are not a result of weak cryptoalgorithms as the details of cryptography are largely beyong the realm of protocol function. Instead he argues that protocol failures are the result of cryptoanalysis of the protocol itself.

In Simmon's work [28] he gives an example of the TMN protocol failure. TMN is a key distribution protocol that was designed to make it possible for a pair of subscribers to an open communication channel in this case a mobile network to end up in possession of a common session key known only to them. The protocol utilizes one unconditionally secure and one provably secure cryptoalgorithm but as Simmons demonstrates it fails catastrophically. It is broken within the time required to set up a secure channel on the network, and the breach is concealed from the server. The idea is that the server carries out the heavier of a pair of computations required to complete the protocol, while each terminal does an instance of the smaller computation. Two terminals C and D are able to collude and trick the server into supplying

a modular cube root of a value whose cube root neither subscriber terminal could extract.

The protocol fails because the session key and the one-time key needed in the key exchange, both from a specified range according to a known probability distribution are assumed to be random but it turns out it is possible for parties to pick them in a manner that is not random. Simmons drives home the point that in any protocol that requires random values it is essential to consider what would happen if values presumed to be random were secretly shared amongst some participants. Ultimately it is important for protocol design to not assume anything that can neither be enforced nor verified. The above case demonstrates an example of a protocol that fails while using strong cryptoalgorithms. In the next section we now take a brief look at other protocol attacks

E. Attacks on Protocols

There are several classes of attacks on protocols. Boyd et al[1] discuss several of them in detail. The adversary's goal here is to try and cause the protocol run to result in a session key to which they are privy. An example is the class of attacks known as *freshness* attacks. Freshness attacks are those where a message component from a previous protocol run is recorded and replayed as a message component in the current run of the protocol. e.g.

3. $Z(A) \rightarrow B : Kbs(K'ab, A)$, here z impersonates A to B and he knows $K'ab$ from the previous run

4. $B \rightarrow Z(A) : K'ab(Nb)$

5. $Z(A) \rightarrow B : K'ab(Nb - 1)$

Another type of attack is the *type flaw* attack. In a type flaw attack the message components are represented in a bit sequence other than what the recipient understands, leading to misinterpretation. An example is switching the nonce and key value order as below in the Andrew secure RPC protocol.

The desired run is:

1. $A \rightarrow B : A, K_{AB}(N_a)$

2.$B \rightarrow A : K_{AB}(N_a + 1, N_b)$

3. $A \rightarrow B : K_{AB}(N_b + 1)$

4. $B \rightarrow A : K_{AB}(K'_{ab}, N'_b)$ where K'_{ab} is the new session key and N_b starts the new sequence for future communications.

The attack is as follows:

1. $A \rightarrow B : A, K_{AB}(N_a)$

2.$B \rightarrow A : K_{AB}(N_a + 1, N_b)$

3. $A \rightarrow Z(B) : K_{AB}(N_b + 1)$ where Z is now impersonating B

4. $Z(B) \rightarrow A : K_{AB}(N_a + 1, N_b)$ now Z replays message 2 to A while impersonating B

Assume that the nonce and the session keys are of same length such as 64bits, A now accepts nonce value $N_a + 1$ as the new session key. Since the nonce does not have to be random, a security problem arises if the nonce is a predictable value.

There are several other attacks on protocols including *Parallel session* attacks where two or more protocols occur concurrently with an adversary using messages from one protocol run to form messages for another protocol run. Improper use of cryptographic algorithms will cause implementation dependent attacks such as *chosen plaintext* attack and *chosen ciphertext* attacks where the adversary uses the system as an oracle to discover the cryptographic key. Another form of attack is one known as a *binding* attack where an intruder Z may convince system users that his public key is the public key of another principal X on the network and subsequently users wanting to send messages to X will unknowingly send them to intruder Z. To prevent this scenario, public key distribution centres require a verifiable binding of public key to corresponding principal.

Kelsey, Schneier and Wagner[32] draw an interesting attack scenario that involves protocol interactions. In what they name as the *chosen protocol* attack, they describe a scenario where an attacker may write a new protocol using the same key material as a target protocol to interact with the target protocol in a way that will cause it to fail. This attack speaks to a fundamental problem in security which is that security is not compositional. The authors claim that for any two systems that are secure independently, there is a way they can be composed that will result in an insecure system. This composition is done via protocols and how protocols interact determines whether or not security is preserved. Two protocols P and Q are said to interact when information received from P can be used by an attacker to successfully attack Q. Often protocol interactions may be facilitated by sharing of key material such as in the re-use of certificates due to costly certification process.

As Lowe[15] observes, many attacks on protocols are very subtle and taking advantage of very slight mistakes. It is this subtlety of attacks and the repetition of mistakes that underscores the need for very systematic development and analysis of protocols. In turn, the need for systematic development and

analysis of protocols requires a framework and perhaps with tool support to attain the rigour necessary to capture the subtlety of security.

F. Security of Protocols

Security of protocols is evaluated against the goals of the protocol. A protocol is good if it achieves its goals. More formally, Diffie, Oorshot and Wiener[7] describe a secure protocol by first defining an insecure protocol as follows:

"A protocol is insecure if any party A executes their part faithfully and accepts the other party B while either B's partial or full record does not match A's or the exchanged key accepted by principal A is known to another party other than B whom A accepted".

In other words, the protocol is secure if the conversation is confidential and records match until the point where A computes and goes into accept state and also if it is computationally infeasible for exchanged keys to be recovered by a third party. Under this definition, trusted server protocols are not secure. In practice though, protocol design entails also giving a security proposition in the context of the specific protocol. For instance in trusted third party protocols, the fact that the server also knows the key does not mean the protocol is insecure.

Here we discuss some general properties used in proving security of protocols. Firstly, a good protocol has the property that it *guarantees the origin of a message*. The origin of a message can only be guaranteed if the message has not changed since it was originated. Use of encryption, digital signatures and hash functions aids in ensuring integrity. Pavlovic[25] presents two protocol proving axioms arising from this property:

First is the origination axiom (rcv) that says if some message t is received by principal A then there must be another entity X such that X originated and sent t and subsequently it was received by A:

$((t))_A \implies \exists X. << t >> \underset{X}{\rightarrow} > ((t))_A$ (rcv)

and the encryption axiom (enc) that if something is encrypted under some principal X's cryptographic key, if it is later decrypted then it must have been decrypted by that principal X.

$A : (\nu x)_A > << E^B(t(x) >> \underset{A}{\rightarrow} > << x >> \underset{x}{\rightarrow}$

$\implies X = A \vee X = B$ (enc)

Another important property of protocols is that *freshness of key* should be guaranteed. There is a whole class of protocol attacks that is based on the adversary gaining knowledge of a key from an old session and replaying it to some principal while impersonating the user with whom it was shared previously and creating a false new session with the unsuspecting principal. To guarantee security, it is important that each user be able to verify that a key gained by the end of the protocol is new and not a replay from a previous session. for this to be guaranteed it is important that a principal have a part in choosing the key value. The user can hence incorporate or otherwise check for a value that they know to be fresh. Such value may be a nonce generated by themselves of some other value such as a timestamp of a counter. For instance if the session key for A and B is a function of their inputs i.e. $K^{AB} = f(N^A, N^B)$ then from A's view once she chooses a freshness value N^A, B cannot choose N^B such that K^{AB} is an old value. The freshness value is often encrypted and sent as a challenge and a correct response provides entity authentication. Pavlovic[25] formalizes the freshness proving axiom as follows:

$(\nu x)_A \wedge x \in Fv(a_B) \implies ((\nu x)_A > (a_B)) \wedge$

$A \neq B \implies (\nu x)_A ><< x >>_A> ((x))_B > a_B$ (new)

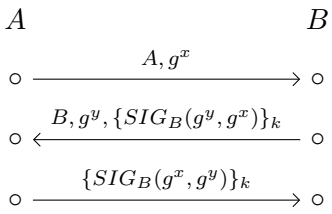
An additional property of protocols that is important but not directly used to prove security is efficiency. A protocol needs to be computationally efficient such that an algorithm can be developed to implement it reasonably. As such one way to optimize a protocol is to minimize the number of message rounds involved in a given run. Where a round refers to all messages that can be sent in parallel from one side to the other. As Bird et al' 92 [20] discuss a good protocol should also have as few computations as possible and be useable on any processing base e.g smartcards and personal computers with no specialized processing chip. It should also not be restricted to any particular cryptographic algorithm. The protocol should function well with any cryptoalgorithm be it RSA, DES or any other. A good protocol should also allow for functional extensions e.g. to accomodate more users or additional messages in fields.

G. Examples of protocols

In this section we present in summary example protocols each for a different purpose. The first protocol is the station to station protocol from the Diffie, oorshot and Wiener discussion of Authentication and authenticated key exchanges, to demonstrate a canonical implementation of authenticated key exchange as envisioned by the authors in those early days. We then proceed to presents discussion of the interlock protocol by Rivest and Shamir as an example of how to expose an eavesdropper. We then present a discussion of the needham shroeder public key protocol because of its historical significance, and we

discuss why it gained significance originally and how it was later found to be flawed and then later on still was modified to make it more secure. these examples will usher us into discussion of protocol analysis.

1) The station-to-station protocol - Diffie et al[7]: The station to station protocol(STS) consists of Diffie Hellman key establishment followed by an exchange of authentication signatures.



Each party forms his own signature with his own exponential listed first because in some implementations it is possible for messages to cross and parties not agree on who sent their message first. The desirable properties of STS are as follows. STS uses challenges rather than timestamps. Authentication is direct in this protocol since both parties encrypt their signatures using exchanged key. Direct authentication means that the authentication goal is achieved at the end of the protocol run without the need for the parties to prove knowledge of the secret key by using it.

Another desirable property of STS is that it uses Diffie-hellman key exchange, hence perfect forward secrecy is achieved as there is no long-term keying material. Additionally the use of DH key exchange means there is no need for a trusted third party. In DH each participant generates their copy of the secret key hence there is no need to transport it.

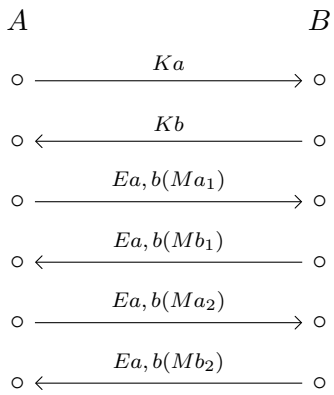
The protocol is weakened when encryption of signatures is removed since exponentials are public and any other party could have signed it if there is no encryption. The protocol is also weakened if each party signs only their own or only the other party's exponential because this exposes the signer to an adversary that could want to use them as a signing oracle. Further, the protocol is also weakened by removing diffie-hellman parameters from the certificates as this allows Eve the intruder to modify Alice's 1st message. Finally, the protocol is also weakened if key exchange and authentication are uncoupled because this makes the protocol subject to man-in-the-middle attacks.

2) Interlock Protocol - Rivest and Shamir: This protocol is set in a public channel with no trusted third party. The goal is to force the eavesdropper to reveal his presence by substantially modifying or

garbling the communication should they attempt to interfere with it.

The adversary C's goal is to mediate between principals A and B by substituting their keys KA and KB with his own KC such that each of them speaks to the other via the intermediary and without knowing it. C can modify key related information.

The security proposition of the interlock protocol lies in A's apriori knowledge of B's communication patterns and vice versa being used to expose the eavesdropper. It assumes that A and B have previsously exchanged their public keys so that C cannot impersonate either of the two parties to the other. The desired protocol run is as follows:



the messages are split into two parts and exchanged interleaved. The transmission of the first part of the message binds the sender to the final cleartext, hence they cannot change it later even though they discover the correct message.

3) Needham Schroeder Public Key protocol - Needham and Shroeder: The original work of Needham and Schroeder[24] is considered a classic in protocol literature. In it they sought to use encryption to achieve authenticated communication. Their original protocol goals were to establish interactive communication between principals, achieve authenticated communication as well as signed communication to guarantee integrity of content. The protocol run is as follows:

A $\rightarrow$ S: A, B, N_A

S $\rightarrow$ A: $\{N_A, K_{AB}, B, \{K_{AB}, A\}_{K_{BS}}\}_K AS$

A $\rightarrow$ B: $\{K_{AB}, A\}_{K_{BS}}$

B $\rightarrow$ A: $\{N_B\}_{K_{AB}}$

A $\rightarrow$ B: $\{N_B - 1\}_{K_{AB}}$

The protocol assumes perfect cryptography, that the adversary has control of the channel and that each principal has a safe environment in which to compute. It is possible for the intruder to impersonate the authentication server by distributing fake public keys. Encryption is used for integrity i.e. to assure A that the KB is the true public key of B from the authentication server.

III. Protocol Design and Analysis

To achieve a good protocol, it is important to outline precisely what should be the system state and what information is gained at the end of a successful run. Several techniques exist for describing and for checking correctness of protocols.

A. Methods of Protocol Description and Verification

Formalizing protocol description and verification is an approach that seeks to solve the problem that informal arguments for protocol correctness are not reliable. Such informal arguments often lead to loopholes going unnoticed in protocols and being exploited later by attackers. broadly, there are two types of formal methods for protocol analysis; Model checking and theorem proving.

Model checking this involves the automatic analysis of systems. In model checking one considers a large but finite number of protocol behaviors. Model checkers enable checking against a set of correctness conditions. This system is good for finding attacks rather than proving correctness hence the reason why it is easier to automate. The cost effective nature of this verification method has made it popular for integration into conventional design methods as a quality assurance standard. It takes in finite state descriptions of the system as well as some expected properties of the system and then outputs a confirmation that the properties hold or otherwise points to some loophole. Model checkers are great for checking on basic standards sometimes reffered to as sanity checks applied to the abstraction of some system. These would include questions such as if an intended protocol run is really achievable within the system and if deadlocks can be reached. However while model checkers easily provide a counterexample for incorrect protocols, they are uninformative when a protocol is correct as they do not provide a reason. Another disadvantage to model checkers is that they are more suited for a scaled down version of a protocol.

We present here a brief evaluation of three arbitrarily selected model checkers. The first is $Mur\Phi$ [8] which was inspired by the *Unity* formalism and has similar properties to pascal statements. It consists of data types, invariants, arrays, global variables, transition rules among other elements that somewhat

combine the properties of a programming language with a finite state machine evaluator. $Mur\Phi$ is non-deterministic hence the execution path can be different as different rules are selected during the processing. The rules change the model system from state to state and at the end $Mur\Phi$ generates all reachable states of the system using a breadth-first search technique. The strength of $Mur\Phi$ is that it does explicit state verification which results in stopping the execution should an error statement in a rule be executed. Also if a state is reached that returns false on the loop invariant then $Mur\Phi$ and returns an error message This behaviour greatly supports debugging. $Mur\Phi$ is uninformative when the protocol is correct and offers no proof of correctness. $Mur\Phi$ is also not optimized for cyberphysical scenarios i.e. to consider the multiple channels in the cyberphysical protocol.

Another model checker is Failures Divergences Refinements(FDR) checker developed by Lowe[....]. This checker checks the process algebra CSP in which communication is modelled by the notion of channels. Each principal is modelled as a CSP process together with its steps in the protocol. The model checker checks for refinement between CSP processes and the property in question. If this refinement does not occur then then an attack can be constructed on the protocol. In this way FDR checks for errors in systems and was first applied to find an attack on the Needham Schroeder Public Key protocol. The limitation of scope for this model checker was solved when Lowe proved mathematically that an attack on the smaller protocol implies an attack on the general protocol. Again one downside to this model checker is that it is geared towards finding attacks rather than producing proofs, additionally it is also two dimensional ideal for cyberprotocols and not fully stretched to explore cyberphysical channels.

Finally we also explore*Brutus* which was designed for cryptographic protocols. This one was designed to provide an protocol checking algorithm that is efficient in space and time and also a tool that could handle multiple protocol sessions in order to expose more complex failures that often only showed up within the dynamics of the live system. The algorithm of BRUTUS explores the state space using depth-first search. If the algorithm encounters a state where the property being checked is not true, it stops and demonstrates a counterexample. Clarke et al [5] describe this system as being completely puch button in nature and its output is an attack or a counterexample demonstating a flaw where the protocol is incorrect. Again this means if an attack is not found the protocol is assumed to be correct since just as any model checker by definition it does not provide proof when the protocol is correct. Like other model checkers and verification systems, Brutus utilizes a first order logic system with well defined semantics that describes the types of messages sent e.g keys and nonces, the participants and the state change rules.

However it is also not optimized for cyberphysical system analysis.

NRL Protocol Analyzer lies between model checking and theorem proving since it has both the characteristic of a model checker that enable it to find flaws and also those of a theorem prover as it can prove security properties of a cryptographic protocol. This analyzer specifies the system state, protocol rules and rewrite rules which are inherited from the Dolev-Yao Model. However it differentiates from the Dolev-Yao model by a wider applicability since it is a general procedure rather than a set of algorithms. Meadows [....] describes a protocol as a form of algebraic system that can be modified by the intruder and this system studies the eefect of such modification. In NRL analysis begins from an insecure state, checking to see if that state can be reached from the start state of a system. If an insecure state is reachable using the given protocol, then an explicit attack can be constructed for it. The NRL protocol analyzer goes a step ahead of model checkers because it provides the ability to inductively prove certain properties e.g. that an intruder cannot learn a word of a specified formal language. It however was not designed for cyberphysical space so it is not clear how it captures the dynamics of cyberphysical computation.

Theorem proving considers all possible protocol behaviors and allows checking against expected protocol behaviors. This system is suited for proving correctness of protocol rather than for finding attacks and it is less automated. Theorem proving provide symbolic reasoning for claiming that a protocol is correct. Their limitation is that they are largely manual and need a lot of human work. An example of a theorem proving system is BAN logic[30]. BAN logic was designed for proving cryptographic protocols. The basis for the logic is belief in the formulas. Correctness is proved based on assumptions also known as rules. In BAN logic the principals, keys and nonces form the primitive working objects. Messages are expressed as a formula of the logic. If an unreasonable assumption is found in a protocol process then it suggests a protocol flaw. The algorithm of applying BAN logic involves determining the goals of the protocol, then determine the assumptions which refers to as an example a party believing in the key they receive from another party and hence being willing to communicate using it. The analysis is done by developing the beliefs up from the original assumptions adding new information as more messages are received. Where the analysis stops, one can check if an attack is possible based on the current messages and current beliefs. BAN logic being a belief system has the ineherent weakness that it cannot handle explicit mathematical modelling as is needed to provide precise proofs although it has its own symbolic language. Additionally as Wessels[30] notes in his analysis, it cannot handle multi-role attacks. There is also the need to extend the framework when handling identities of principals and participants, something

that takes away from the intended simplicity of the model.

B. Theorem Proving and Formal Approaches to Protocol Analysis

While there has been work on writing protocol proofs, a big part of protocol analysis is attempting to design attacks on protocols and for as long as an attack is not found on a given protocol it is assumed to be secure. We take a look now at the work of Fabrega, Herzog and Guttman in strand spaces[6] where they attempt to directly prove protocols correct rather than by finding flaws. This framework provides for more mathematical rigour. In this system message sends and receives including those of the adversary are represented as strands. Elements of a strand are called nodes and the set of all possible strands is the strand space. Properties are proved by induction on well-founded sets. This model takes the possible behaviors of a system and then attempts to put bounds on the abilities of a intruder. Various notions of security are addressed including secrecy and authentication. This model attempts to give informative proofs with information such as conditions for reliability of a protocol using specialized term algebra and other diagrammatic elements of the concept of strand spaces. The algebra of strand spaces appears to be designed for cyberspace two-node protocols although it seems promising for extension to the cyberphysical process which involves a lot of physical factors incident to the environment where the protocol runs.

Some early work in protocol verification is due to Dolev and Yao[22]. In their work on the security of public key protocols they discuss that public key systems are good against a passive attacker e.g an eavesdropper but then once we have an active attacker trying to manipulate data, the design details of a protocol become very important. As early as back then they recognized the subtlety of attacks and the need to replace informal arguments with formal arguments for protocol security. Their model is defined as follows:

”Let f[1],...,f[r] be a two party protocol between a sender A and receiver B. The protocol is insecure if and only if for the honest parties the adversary has access to a sequence of functions g1,...,gk such that gk o ... g2 o g1 o f[1]A,B = id.”

One of the strengths of this model is that it makes minimum assumptions about the adversary and with the only security goal that an active attacker intruding on communication between Alice and Bob should not recover their secret message M. The model assumes public keys are easily accessible and also that the adversay can start concurrent sessions with both principals. One of the limitations is that the honest

parties are stateless and cannot store information they can use later while the adversary is able to store information. However this is a practical approach. Another limitation is that it is complex and may be harder to correctly implemen. A simpler approach to the model due to Dolev, Evan and Karp[21] exists still on the ping-pong protocols i.e. protocols where parties cannot use information collected from previous messages due to their being stateless.

This latter model starts with a set of all words over the alphabet Ea,Eb,Ez, Da,Db,Dz,ia,ib,iz,da,db,dz,d that simplify to the empty string using the cancellation rules Dx Ex = Ex Dx =dx ix = d ix = epsilon. This set of words is context free and the grammar can be converted into an equivalent Push Down Automaton of constant size not dependent on the protocol. From the PDA one can build and nondeterministic finite automaton accepting all the strings of the form gk o ... o g1 o f[1]A,B where each gk is one of the finitely many functions the adversary has access to. Combining the PDA and NFA using a cartesian product construction a new PDA can be obtained that accepts the intersection of the two languages and the next step is to decide if the language of a PDA is empty or not. The above two models are mainly built to provide cryptographic proofs of security. They are both designed for two-party protocols. Because the above models are not very expressive, researchers continued to extend them to achieve a middle ground between on one hand cryptographic correctness and formality that can be moulded into a tool and on the other hand some expressiveness somewhat providing accessibility to the model.

A good example of a model checker extending from Dolev-Yao while increasing expressiveness is the Maude-NPA[9]. Maude-NPA also the work of Meadows is an advancement of the NRL protocol analyzer(NPA) discussed earlier[17]. NPA inherits directly from Dolev-Yao particularly the technique of rewrite rules. As mentioned earlier NPA differentiated from the Dolev-Yao model by its wider applicability since it is a general procedure rather than a set of algorithms. However in the NPA the theoretical basis of the rewrite rules and narrowing was not covered in depth. In Maude-NPA rewrite rules and narrowing is developed with a deeper theoretical foundation which then allowed it to be able to handle a wider range of algebraic properties of cryptographic systems such as encryption-decryption cancellation properties, Diffie-hellman exponentiation and Abelian group properties among other algebraic properties of cryptosystems. The tool which is described by the authors [9] as an inference system for reasoning about cryptographic protocols is applied both for finding attacks and providing proofs. Its underlying mathematical model is strand spaces described earlier in this section while the system is written in Maude which supports equational and rewriting logic. This system is quite advanced and the only shortcoming

that comes out from a cyberphysical perspective is that it is not currently made for analysis of interactions in cyberphysical space.

In the direction of cryptographic security, Bellare-Rogaway[3] also worked on a model to prove security by first assuming an adversary can obtain significant advantage and use this to check if the encryption mechanism provides indistinguishability. If encryption is secure, the adversary gains no advantage by accessing the messages. The Bellare-Rogaway model uses a reductionist proof approach to analyze what the adversary can do, how the adversary can break the protocol. The model then reduces the problem of the adversary to some known hard problem thereby demonstrating that if the adversary can break the protocol, he can also solve some intractable computational problem. Itś stength is that it is applies a pessimistic view of the adversary, modelling a powerful adversary that controls the communication channel being able to initiate protocol runs between principals in the form of oracle queries, being able to alter messages and even corrupt principals to obtain past session keys. The adversary is however bounded computationally to probabilistic polynomial time. A protocol is said to be secure if any intrusion an adversary can perform is benign i.e. a principal only accepts messages from the adversary if they are merely a relay of otherwise good messages from a legitimate principal that is party to the protocol. The limitation of this model is that it was designed for the symmetric, two-party setting. Subsequent to this 1993 work there has been other works extending this model for various settings including server-based protocols, key-agreement protocols, password-based protocols and group-key agreement. The general limitation is the niche focus.

Woo-Lam [31]present a protocol execution as a finite alternate sequence of global states and transitions where there is a start state, final state and intermediate states all corresponding to some action. The protocol specifications are designed to resemble programs with an underlying algebraic specification. Their model is concerned with secrecy and correspondence as the measures of correctness while ignoring other aspects such as peer awareness of key. Kemmerer et al[33] also use the state machine concept in their interrogator protocol analyzer that is designed to take a formal protocol specification and look for message modifications that are not consistent with the protocol's goals.

The formalization work we begin subsequent to this document picks up from the framework known as *protocol derivation* logic due to Cervesato, Meadows and Pavlovic[4]. In this framework, protocol fragments are composed and patterns matched to properties. The protocol participant in this case observes some

local events and makes some deductions about the peers in the protocol. The system enumerates properties supported by the protocol based on its construction. When an expected property does not manifest, this indicates a missing component or component failure and a counterexample can be constructed. In this framework there are protocol taxonomies to help in understanding, choosing and devising protocols based on desired features. The framework was designed for cyber-based protocols.

IV. CYBERPHYSICAL SYSTEMS

To motivate our main subject which is cyberphysical protocols, we now present a brief and introductory overview of cyberphysical systems. Cyber-physical systems (CPS) are physical systems that are controlled, monitored, or otherwise manipulated by a computer [27]. It can be summed up as a confluence of cutting edge communication technologies [12]. Rajkumar et al give a good summary of the enabling properties of these high end technologies and they include; low-cost and increased-capability sensors with form factor trending downwards; lowcost, low-power, high-capacity, small form-factor computing devices; increased reliability of wireless communication compounded with greater internet bandwidth; improving energy capacity as well as alternative energy sources. All these serve to reduce the cost and improve the functioning of cyberphysical systems. However as they mention, the challenge does lie in interfacting the precision of cyber-computing with the uncertainty of physical environment to achieve a co-ordinated system with a logical flow of information. Here the word cyber is derived from the word cybernetics which defines networked communication, computation, and control[12].

The core of the functioning of CPS systems are the sensors and actuators. The computing units are fed input by the sensors which are able to detect various changes in the physical environment of the systems and these are communicated via the cyber network. The computational units then estimate the system state and generate appropriate signals for the systemś actuators. Humans become part of the CPS both because their actions in the physical environment may be detected by the sensors and processed and thereby cause a system state change and also because humans can override the system functioning and enter commands to cause a desired change in state through the human computer interaction interfaces. The CPS reorganizes and reconfigures dynamically and with a high level of automation in the flow of control. The systems are often highly complex and with a high level of coupling between the physical and computational components.

The referenced discussions on CPS [12] and [27] are in a futuristic tone and they go on to identify the

nuts and bolts that underly the success of cyberphysical systems. One challenge is the need for advanced software and hardware authentication systems. This is in the light of possible interference of signals both along the cyberchannels and on the physical distance between various components of the CPS during the process from sensor to processing to sending of signal to actuators. Often any interference with have critical repurcussions e.g. in remote robot-assisted surgeries. Other challenges include the design of crossplatform technologies for various real-time operations, automating safety and security failover in critical CPS as well as decentralization of control.

As [12] point out, there are various tools and frameworks currently available to support modelling of control systems. These include Matlab, Simulink, Labview and others and they work well for engineered systems but do not scale up to cyberphysical system design modelling. One of the various issues that make CPS control different from control of purely engineered systems is that in the CPS there is user intent as a factor. CPS need to adapt to user intent as relates to function of the system. Not only in terms of speech patterns or pressing of the correct buttons as would be in a fully engineered system, but also in terms of the possibility that the user is attempting to misuse the system or the system is expected to respond to other more complex things like the sound of crying or screaming. There are also other scenarios such as CPS that need to perform certain actions based on facial recognition or recognition of mood as an example.

Here we briefly describe the various directions in cyberphysical systems technology. In a 2010 report, the department of homeland security [1] identified major directions for CPS security based on sectors that heavily utilize cyberphysical technologies.

1) Energy

   By 2010 the electric grid had nearly 1,000,000 megawatts of generating capacity being distributed to about 250,000 transactions per day on the power market. The electric grid management is complex and relies on systems such as Remote terminal units, SCADA systems and other systems. These are susceptible to environamental interference from natural disasters e.g. lightning as well as man-made accidental and malicious disasters. The electric transmission and distribution network needs to be security smart inorder to respond to and mitigate local disturbance, receive commands in response to a global incident, understand and monitor its own state and be able to communicate its state information within and across utilities. The ideal electric CPS would be smart to identify outages

and surges, resilient to contain outages by rerouting around failure points, flexible to accomodate off-grid power alternatives, reliable to manage load balancing and secure by being less susceptible to harm. smart electric grid would also be more interactive to the customer. These dynamic interactions and the general connectivity are all part of the security threat to the electric grid.

2) Water

The main activities involving water systems are clean water distibution and waste water collection and the treatment of both streams. These rely on the SCADA and the DCS systems for distribution and collection and for treatment respectively. Water systems make use of wireless communications as well as telephone communication, internet and wire for various control actions such as linking the monitoring system and controls for the treatment and distribution systems to a central display and operations room. Due to this cyberphysical interconnectivity ad well as other physical problems such as aging infrastructure and accidents, the water system is also susceptible to various forms of vulnerability including internet worms and viruses. Water systems should also be able to self-assess, mitigate local interference, co-ordinate with other systems to respond to widespread incidents and generally be less susceptible to harm.

3) Transportation and Aerospace

In Aerospace as a transpostation example, the airplane itself is a CPS which can compute its location using geograpical positioning system(GPS) and communicate this information to air traffic management(ATM) centres on the ground in real-time. One of the important aspects of aerospace is the adhoc networks for airborne aircrafts to communicate to ground and to other airborne planes. open networks threaten the security of aircraft control and also of the data being transmitted. Challenges include how to balance security and safety in a unified model e.g. secure yet efficient handling of safety-critical avionics software updates. There's also a challenge to quantify the threats from cyber-world on the physical space of the aircraft as well establish an effective system of managing the human in the loop. The aircraft also will with time have failures and replacements of subcomponents which raises the issues of reliable integration. There are various other issues including passenger privacy.

4) Chemical Sector

The chemical sector within itself is diverse spanning from basic chemicals such as household soap and detergents, to specialty chemicals such as lab experiment chemicals as well as agricultural

chemicals, pharmaceuticals and other products. Within the chemical industry the security requirements are for information and process control. The chemical grid is controlled by industrial control systems (ICS) and distibution control systems(DCS) for chemical production and manufacturing as well as SCADA systems for monitoring and supervisory control in the pharmaceutical and petrochemical industries. In the chemical sector supervisory control is limited to a small area at a time, typically the physical boundaries of a production plant. However there is interconnectivity among engineers, vendors, maintenance personnel and other involved parties. Therefore chemical processes are still susceptible to attacks such as denial of service and others that may interfere with process control remotely and cause leakage or other undesired behaviour of material that may be hazardous. The chemical industry is however slightly better than other sectors in the sense that its damage is likely to be localized due to deliberately minimized connectivity. The goals in this sector include developing comprehensive self-vulnerability assessment tools for chemical systems facilities as well as authentication and insider threat control at facilities.

5) Healthcare and Public Health and Medical Devices

   Medical devices are of many forms including those that may be used to provide treatment and medical care as well as manage and communicate patient records electronically as well as devices implanted in human bodies to provide various forms of health support. Dangers to medical devices include the risk of tampering by malicious or accidental reprogramming of firmware and since many of these devices are connected to the internet, such tampering can occur remotely. Also data in transit may be intercepted and copied by malicious parties and also it may be altered or replaced with fake data. Control data travelling remotely from device to device or personnel to device may also be at risk of tampering. Security goals in this sector include acheiving timely detection of security violation and abnormal behavior of devices, authentication of interacting devices and validation of security in integrated systems.

6) Commercial Buildings and Facilities

   Here the CPS includes monitoring and control of the building conditions and attributes as well as interactions within the building. These conditions and attributes incude air contitioning, automatic doors and security alarms while the interactions may include the use of wireless communications within the building. The control network includes smart sensors and actuators that support system functioning, remote control and other processes. Building automation systems are susceptible to

denial of service attacks which can have critical effects on building operations or even lives of occupants. Challenges include verification of interaction components of the network, safety and security analysis in a unified model

7) Law Enforcement, Security and War Systems

In law enforcement and defense, CPS applications include smart guided weapons and defensive wearable computing. Other CPS include intelligent unmanned vehicles and supply chain logic systems as well as emergency response systems [29]. These systems also face the challenges of susceptibility to denial of service as well as control data from unauthenticated sources.

1) Cyberphysical Security Challenge: Cyberphysical structure presents an extended set of challenges as NIST discusses[29]. The combination of cyber and physical vulnerabilities may itself present a new and harder to analyze, higher impact risk in critical systems. Protocols are necessary to ensure optimal physical and computing network interfacing, composability and interoperability that maintains normal behavior of components in cyberphysical systems. The challenges are manyfold including managing the human in the loop who has more impact on the cyberphysical system function than the would on a purely engineered system. Additionally the environment is often multidisciplinary since computation occurs within highly specialized environments such as medical or air traffic control. In their 2013 call to action, NIST[29] indicates that future CPS will have many sophisticated, interconnected parts that must instantaneously exchange, parse, and act on detailed data in a highly coordinated manner. The goal is to achieve timely output, outcome agreements, resilience, data transfers and technical seccurity through seamless and efficient protocols. CPS should also eventually have standard governance models for structured control and regulation to mitigate vulnerability and consequent liability.

V. Cyberphysical Protocols Research

A motivating preliminary work is that of Ellison[9] mentioned earlier in which he discusses the concept of a protocol as a 'ceremony'. The protocol is termed as a ceremony because it is a distributed system involving the human node and hence has various channels of communication involved other than the cyber channel. For example the visual and audio channels of communication. The key takeaway from their discussion is the significance of the human node in a lot of distributed systems and protocols vs the difficulty in analyzing it. Traditional protocol analysis leaves out the human node as an out-of-band element in the process. Perhaps this is so because of the simplicity gained in not interrogating how for example a client computer comes to be in possession of a PKI root key and be sure that this is not

compromized. The difficult in analyzing the human node arises on several levels as pointed out in that work. As an example the HTTPS ceremony involves first provisioning the client computer with the PKI root key which happens out-of-band as far as HTTPS protocol is concerned.

This protocol assumes server authentication and use of RSA public keys. The cient and the server are both set to some state at the start of the protocol whereby the client knows a root key that certifies key pairs and an address to which it wants to communicate. From a ceremony perspective though, the provisioning of PKI root key is also subject to analysis. As described in Ellison's work, the root key provisioning process relies on a concatenation of channels. The certifying authority delivers its root public key over a channel to a human say denoted as R. Human R delivers this key over a channel to a user, X. The user delivers that key to her computer. All three channels need to be secure for the process to be secure. This analysis is indeed more complex than the one of the protocol above because the security of human-to-human channels in the process depend on various factors affecting them, their pre-existing state, what information they have about about the certificate chain as an example and how they authenticate each other on a human-to-human level and authorizing access to each other's computers. These factors need to be analyzed in order to say that a process is secure. There are various issues that make it more complicated to analyze the human node. For instance once we add the human node to the protocol, a lot of association arise that need to be represented. e.g how individuals and components on the network that are not supposed to be communicating are associated with each other. Ellison's gives an example among others of a phishing url existing to lead users astray while they try to reach some service on its legitimate website. One question in this example can be how to model user decision making and what can be done to reduce the chance that they will click on a fake url. Perhaps the formalism of actor networks can offer some tools that can be used to analyze protocols in this cyberphysical way. The concept of actor networks can be traced back to the work of Latour[16] and possibly further back if you consider the 1970s work of Hewitt et al[14] who used the actor formalism within their work on artificial intelligence and concurrent computation albeit in a somewhat different perspective. Hewitt called actors the fundamental unit of computation embodying processing, storage and communication. In their work actors refers to processors and as actors they can have subactors or they can combine and form bigger actors, start processes which in turn can start other processes specify what is to happen with their internal and other processes.

Closer home, Latour describes actors as elements in the network that have the property of agency. These

may include inanimate objects and humans. In that regard he gives an example that "A billiard ball hitting another one on the green felt of a billiard table might have exactly as much agency as a person directing her gaze to the rich human world of another meaningful face in the smokefilled room of the pub where the tables have been set up." He opines that 'social' for an actor network is a momentary association characterized by the way the network gathers together into new shapes. In his view the actor is an object of interest to another entity which then finds a way of connecting with it to form some momentary association for a specific purpose. One of the things Latour and Hewitt works have in similarity is that an actor itself is a network that can have a channel to itself to paraphrase Hewitt's words[14] and Latour's that an actor is the centre around which momentary social structures are continually formed and broken but also an actor has agency to itself. Latour's work is however largely from a social scientist's point of view and not from the perspective of that gap in analysis of the human node in ceremonies as was described by Ellison.

A. Formal Actor Networks and Protocol Derivation Logic

Meadows and Dusko give the human node question some initial concrete treatment in their work on actor networks which begins to look at every participant in the protocol as an actor. However they don't look only at the human node but also every object node inanimate or not that is present and participating in a network. This is particularly relevant in today's world where cyberphysical systems are a dominant component of computing as a whole. What Meadows and dusko do is begin to model mathematically this formalism of actor networks. In their work too an *actor* is a participant in a protocol. An *actor* is represented by a set of nodes A where a node can be atomic representing a complete entity by itself.

ANts have a tree structure. The set of nodes in an ANT have a certain parent-child or sibling hierarchy between them which can be represented in a tree. At the root of the tree is the entire process which branches down to the various actors and configurations that are part of the process. A configuration will then branch down to the component actors as its leaves. Actors can also branch down to their respective subactors.

Since ANts have a tree structure they also have a graph structure. The set of nodes in an ANt are interconnected by various channels via which they can communicate. The nodes correspond to vertices denoted with V in a graph structure, while the channels are drawn as edges denoted with E connecting the nodes. An ANt graph is directed since information flows in some direction. In modelling actor network

procedures, these authors apply the protocol derivation logic formalisms first used in their work in deriving the GDOI protocol[4]. In that work they construct protocols incrementally adding new functionalities and composing systems as necessary and proving them as they go. In this model as features are composed the properties add up while preserving each other's assumptions. They also construct the attacks in similar way. This points to the fact that ANts can compute by changing self.

We know that in cyberphysical space various network changes occur during the run of a protocol such as objects arriving into or leaving a scene. The modular representation of protocol participants as nodes in actor networks and their ability to combine and form bigger nodes or open up into their subnodes brings to mind the procedures of induction and recursion respectively. Indeed this is one of the properties that make actor networks ideal for modelling cyberphysical protocols. With actor networks it is possible to apply a process algebra as well some discrete logic in reasoning about network computations. This opens a door for designing of a robust verification and proof system.

In Meadows and Pavlovic's work on protocol derivation logic they while focusing on cyberspace introduce a process algebra in which they describe protocol action and interactions. Other authors also developed various syntax for protocol description and analysis including the strandspaces model described earlier in this document. However the work of analysis and verification of protocols has not gone beyond cyberspace to any notable extent to the best of this author's knowledge.One of foundational logics is predicate logic together with related formalisms including hoare logic.

One advantage of using predicate logic approach is that non-determinism can be introduced simply which makes it easy to make specifications in the case of programs and by extension in protocols and this facilitates abstraction [13]. Hoare rules can be summarized up as a predicate representation of assignment of properties to variables, presentation of skips in computation which is the case when nothing happens and hence nothing changes, also a representation of sequential composition, if statements, while statements and statements of consequence.

REFERENCES

[1] Adam, N. *Workshop on Future Directions in Cyber-Physical Systems Security*, Department of Homeland Security, 2010.

[2] Boyd, C. Mathuria, A. *Protocols for Authentication and Key Establishement*, ISBN: 3-540-43107-1, Springer, 2003.

[3] Bellare, M., Rogaway, P. *Entity Authentication and Key Distribution*, Advances in Cryptology-Crypto 1993 Proceedings, 1993

[4] Cervesato, I, Meadows, C., Pavlovic, D. *Deriving Key Distribution protocols and their security properties*, 2006.

[5] Clarke, E., Jha, S., Marrero, W. *Verifying Security Protocols with Brutus*, ACM Transactions on Software Engineering and Methodology, Vol. 9, No. 4, October 2000, Pages 443487.

[6] Cholewa, A. *Maude-PSL: A New Input Language for Maude-NPA*, University of Illinois at Urbana-Champaign, 2015.

[7] Diffie, W., van Oorschot P., Wiener M. *Authentication and authenticated key exchanges*, Designs, Codes Cryptogr 2:107125, 1992.

[8] Dill, D. *The MurΦVerification System*, Stanford University.

[9] Escobar, S., Meadows, C., Meseguer, J. *Maude-NPA: Cryptographic Protocol Analysis Modulo Equational Properties* Universidad Politecnica de Valencia and Naval Research Laboratory and University of Illinois at Urbana-Champaign.

[10] Ellison, C. *Ceremony Design and Analysis*, Microsoft Corporation, One Microsoft Way, Redmond WA, 2007.

[11] Fabrega, F. *Strand spaces: proving security protocols correct*, Volume 7 Issue 2-3, Journal of Computer Security, Pages 191 - 230, Jan. 1999.

[12] Guturu, P., Bhargava, B. *Cyber-Physical Systems: A Confluence of Cutting Edge Technological Streams*, University of North Texas and Purdue University.

[13] Hoare, C. *Programs are predicates*, In Fifth Generation Computer Systems. Pages 211218. June, 1992.

[14] Hewitt. C, *Hewitt, Meijer and Szyperski: The Actor Model (everything you wanted to know, but were afraid to ask)*https://channel9.msdn.com/Shows/Going+Deep/Hewitt-Meijer-and-Szyperski-The-Actor-Model-everything-you-wanted-to-know-but-were-afraid-to-ask

[15] Lowe, G. *Breaking and fixing the Needham-Schroeder public key protocol using FDR.* In Tools and Algorithms for the Construction and Analysis of Systems, pages 147-166. Springer-Verlag, 1996.

[16] Latour, B. *Objects Too Have Agency*, in Reassembling the Social: An Introduction to Actor-Network Theory, Oxford: Oxford University Press, 2005.

[17] Meadows, C. *The NRL Protocol Analyzer: An Overview*, Center for High Assurance Computer Systems

[18] Menezes,A., Qu, M., Vanstone, S.*Some new key agreement protocols providing implicit authentication.* In Workshop on Selected Areas in Cryptography (SAC'95), pages 22-32, 1995.

[19] Diffie, W., Hellman, M. *New Directions in Cryptography*,IEEE Information Theory Workshop, Lenox, MA, June 2325, 1975 and the IEEE International Symposium on Information a different form. Theory in Ronneby, Sweden, June 2124, 1976.

[20] Bird, R., Gopal, I., Herzberg, A., Janson, P., Kutten, S., Molva, R, Yung, M. *Systematic Design of a Family of Attack-Resistant Authentication Protocols*, IBM Raleigh, Watson & Zurich Laboratories, 1992.

[21] Dolev, Evan, Karp, *On the security of ping-pong protocols*, Inform.and Control 55:57-68 (1982)

[22] Dolev, Yao, *On the security of public key protocols*, IEEE TIT 29(2):198-208 (1983)

[23] Merz, S., *Model Checking: A Tutorial Overview*, Institut fur Informatik, Universitat Munchen.

[24] Needham, R., Schroeder, M. *Using encryption for authentication in large networks of computers*. Communications of the ACM 21 (12): 993999. doi:10.1145/359657.359659. 1978.

[25] Pavlovic, D. *Authentication Protocols* teaching notes, Principles of Security, Oxford University, Michaelmas term 2008.

[26] Pavlovic, D. and Meadows, C. *Actor-network procedures:Modeling multi-factor authentication, device pairing, social interactions*, 2011.

[27] Rajkumar, R., Lee,I., Sha, L., Stankovic, J. *Cyber-Physical Systems: The Next Computing Revolution*, Design Automation Conference 2010, Anaheim, California, USA.

[28] Simmons, G. *Cryptanalysis and Protocol Failures*, ACM, Volume 37 Issue 11, Pages 56-65, ACM New York, NY, USA, 1994.

[29] Stipanovitz, J., Ying, S., Cohen, I, Corman, D., Davis, J., Khurana, H., Mosterman, P., Prasad, V., Stormo, L., *Strategic R& D Opportunities for 21st Century Cyber-Physical Systems*, NIST, 2013

[30] Wessels, J. *Applications of Ban-Logic*, CMG FINANCE, 2001

[31] Woo, T. , Lam, S. *Verifying Authentication Protocols: Methodology and Example*,Citeseerx, 1993.

[32] Kelsey, J., Schneier, B,. & Wagner, D. *Protocol Interactions and the Chosen Protocol Attack*, Security Protocols, 5th International Workshop April 1997 Proceedings, Springer-Verlag, 1998, pp. 91-104.

[33] Kemmerer, R. , Meadows, C. , Millen, J. *Three Systems for Cryptographic Protocol Analysis*, Journal of Cryptology(1994) P79-130, 1994.