TOTSCO

End-to-end Encryption Responses *as requested in Bulletin 12*

With developer's guides and software development kits provided by TOTSCo, do you feel confident that you could implement E2EE?	Would this delay your development? Does it affect your confidence around testing and going live?	Do you anticipate operational complications?	Do you think that adding E2EE would impact your ability to meet the 60s SLAs?	On balance, do you think that E2EE should be implemented: a). never b). at go-live c).maybe after go-live	For which development languages would you like to see example of implementation of E2EE to be provided by TOTSCo e.g. Java, PHP, Python, C#	Other Comments
YES we do.	NO. The process of signing, verifying a signature, encrypting a string or de-encrypting a string are a minimal part of the overall work required to integrate against matching, and switch requests. Additionally, they are easy to test, because the E2EE process either works or it doesn't. Not communication with another CP is possible without a working E2EE and so it is easy to detect and resolve if it fails testing. This is unlike the other parts of the protocol which potentially require significant edge and integration testing.	NO. It is anticipated that a suitable design for key management mediated by the hub, should be simpler to manage and maintain than using TLS and certification management	NO. E2EE is a transport level concern and add not additional complexity to the matching process once implemented.	BEFORE-GO-LIVE. End-to-end encryption must be implemented at launch. Given the time and effort that CPs are going through to integrate with the hub, it isn't envisaged that any further round of development, particularly that that would require coordinated testing across the industry, would be acceptable or even conceivable. The benefits of E2EE cannot be over stressed, and a missed opportunity to implement it will be an industry failure.		
Yes, [we] feel confident we are able to implement E2EE with an SDK provided	There may be a delay with development and testing by implementing E2EE due to the keys needed to be shared and readiness of the multitude of CP's to support this. It's possible there are CPs that may need significant platform changes to support	Yes we do anticipate operational complications due to key management for new onboards and maintenance for when keys need to be updated	It's not clear at this stage, but there is an overhead in using E2EE, until we are able to E2E test its difficult to comment at this stage	We believe it would be best to look at this after go-live as its anticipated there will be complications and there may some CP's which need significant changes to their platform to support. Post go live would allow a progressive approach without adding complexity to an already challenging timeline	Development would be implemented via dot.net framework C#	

						[We] agree that there is a need for E2EE but, we do not have a specific preference in terms of how it is implemented. We would however stress the need for visibility of a clear timeline in the decision-making process, particularly if additional requirements are going to need to be built for launch.
This respondent requested confidentiality						
This respondent requested confidentiality						

A dedicated workshop on E2EE with TOTSCo and CPs would help to understand the requirements including the scope of work required on the end-to-end encryption (E2EE) of messages along with implementation of the related message signing function. The above is in addition to dedicated E2EE developers guide and the software development kits to be provided by TOTSCo for implementing E2EE.	[We] support E2EE to secure customer data in transit. Due to change in scope implementing E2EE would require additional development effort and industry testing and this would add to the lead time for go live.	1. Additional overhead for processing public / private keys. 2. Management and maintenance of public and private keys generated by CPs over time can impact customer experience. 3. Business rules are required to be understood around the timeline the historical keys will be maintained. 4. If no public key is available in the TOTSCo directory for the recipient CP then we cannot encrypt the message. This require clarification from TOTSCo if multiple message format are required to be supported which will further increase the scope of work required. 5. For the CPs who deploy managed hub solution how will E2EE will work is required to be understood. How will those CPs encrypt / decrypt the messages.	We don't anticipate any issues with this but end to end industry testing would be required to validate this assumption.	[We] agree and support the E2EE of customer data during transit which will make One Touch Switch solution future proof including protection against any security breaches. This is change in scope and will add significant lead time for development and industry testing but [our] view is that CPs should work towards implementing it with aim for deploying at go-live.	Java	[We] support E2EE for the protection of end customer's data. [We] would expect the requirements including the API spec to be baselined asap including any updates for E2EE. The test harness is required to support messages with E2EE to make it meaningful
Yes	Whilst implementing E2EE would be more complicated than not implementing E2EE in our view the additional time required would not significant	No	No	At Go Live	Any	
YES. The TOTSCo Developer's Guides are simple to follow, and therefore we are confident that we could implement E2EE.	NO. We do not anticipate any delay. More importantly we would expect a faster delivery from TOTSCo because our customer data will no longer be visible to TOTSCo. This change is ultimately beneficial for the entire industry.	NO. As a CP, we use E2EE encryption when necessary to protect our customers.	NO. In our implementation, E2EE has made no measurable impact on response times.	AT GO-LIVE. We must ensure the confidentiality of consumer data and deliver the HUB as fast as possible. E2EE simplify the HUB development and processes while protecting the industry from industry-wide data breaches.	Any	
This respondent requested confidentiality						

The process described is not dissimilar to processes we have already adopted for integrations with other partners and so, with appropriate guidance from TOTSCo, I don't perceive implementation of this to be a problem.	While this will add to the development workload, it shouldn't be a significant element of the overall work and delays in finalising the API are likely to be more impactful.	There will be some inherent additional operational complexity, for instance around the management / retention of key-pairs, but the design parameters and key management policy suggest (6 weeks retention and 3-month key-rotation) are not perceived to be overly onerous, and this is a process we are already involved with for other integrations.	No, we wouldn't see this as significantly impacting the ability to meet the 60S target based on current understanding.	I would prefer implementation from go-live. I suspect that having it implemented after would create additional complexity and inertia that would delay full implementation. Mainly though, I would consider it a necessary function rather than a 'nice-to-have', especially the signing element. Given the risks of industry-wide impacts from a single breach-point at the hub or any single provider, and looking at a likely timeframe for deployment and requirements around TSA, I would much rather have this in place from day-1.	Any - Preference Python	
Yes, based on the draft E2EE specification v0.2, with a common RSA/AES process, we can implement E2EE.	It would prolong the development time and extend the overall timeline. The full impact can't be predicted yet, but it will certainly increase the risk of not meeting the timeline. It will likely make it more difficult to do initial (manual) testing, as requests/responses can't be crafted/read in plaintext. Sending/receiving messages to/from our own implementation, prior to connecting to the Hub, can't be made without tooling including E2EE-support.	Not necessarily, but encryption adds an additional level of complexity, which can make it more difficult to troubleshoot potential issues. In cases of communication/message exchange failures, the cryptographic process is another possible source-of-failure which must be considered and handled. e.g. incorrect/invalid/missing keys etc.	Not in the common case. The cryptographic processes seem simple enough not to affect the processing of a single message in terms of time. In cases where the ultimate recipient may be missing the key used by the sender (for signing), an on-demand request must be made to retrieve the key, which may affect the SLA. This should rarely occur but may occur if a sender has published a new key which they use for signing, which hasn't yet been retrieved by the recipient. In this case, the time required to obtain the key, including the Hub	o We believe that TLS (encryption at transit) should be enough assumed that involved parties trust that the hub is secured. All parties that act as an endpoint must take all necessary measures to secure their data, which E2EE cannot address by itself. o However, if implemented, we believe it should be implemented from start, with the understanding that it may affect the go-live/timeline for all parties § Adding it after go-live, adds additional development and upgrade work	C# (.NET Framework 4.8 and dotnet 6)	Will E2EE, if/when implemented, be mandatory? Or are all parties required to implement and support both unencrypted (using TLS only) and encrypted (E2EE) communication?

			response time, count towards the SLA.			
Yes. There is no doubt that end-to-end encryption could be implemented. Cryptographic assurance and protection of messages is already routine in other systems, and in widespread use in industry, so developers and OSS/BSS vendors are familiar with the fundamental principles and practical use of cryptography and key management.	Yes. There is little doubt that it would require additional work to implement and test E2EE. However, we do not anticipate that it would be a very large piece of work; at worst we believe it would add 4-6 weeks of development effort to fully build and test, and so could be accomplished more rapidly than this in real terms. We do still believe we can implement our TOTSCo Hub integration within timescales that align with TOTSCo's planned delivery dates for the Hub.	No. Within our operational support system, the underlying encryption will be managed. For authorised end-users within our secure portals, data will be decrypted automatically and so transparent for users managing operational processes. There are of course potential issues where message encryption or decryption fails due to incorrect key material being used, stored, or improperly managed (failure to update TOTSCo, etc). These would be managed as faults and thus managed through typical operational processes, just as we would any other fault with switching communications. Key material will have to be managed, along with typical processes for key material rotation, secure storage, and so on. These are processes that are already in place for other key material, so no practical complexity is added.	No. Message processing would incur one extra validation and decryption step on inbound messages, and an encryption and signing step on outbound messages. The overhead per message is essentially nil. While messages that fail decryption could not result in a response in 60s, this is no different to other malformed messages.	Now. The complexity of adding E2EE is small. Managing both E2EE and plaintext messages would require additional development effort, likely doubling the required effort overall. However, this complexity comes with a permanent and ongoing overhead which would not be incurred by mandating E2EE at the outset, as both code paths would require ongoing maintenance and support, unless non-encrypted payloads were subsequently forbidden on the hub. Experience of industry in similar areas shows that retroactive implementation of voluntary encryption schemes is typically slow and incomplete (e.g. RPKI for verification of BGP announcements). Security requirements are ever-increasing as UK service providers and consumers come under increasing attack from threat actors. It makes more sense to pre-empt future regulatory requirements by adopting current best practices.	Python, C#, Java, Rust.	

YES The TOTSCo Developer's Guides are simple to follow, and therefore we are confident that we could implement E2EE.	NO We do not anticipate any delay. More importantly we would expect a faster delivery from TOTSCo because our customer data will no longer be visible to TOTSCo. This change is ultimately beneficial for the entire industry.	NO As a CP, we use E2EE encryption when necessary to protect our customers.	NO In our implementation, E2EE has made no measurable impact on response times.	AT GO-LIVE We must ensure the confidentiality of consumer data and deliver the HUB as fast as possible. E2EE simplify the HUB development and processes while protecting the industry from industry-wide data breaches.	Any	
YES. The TOTSCo Developer's Guides are clear and therefore we are confident that we could implement E2EE.	NO. We do not anticipate any delay. More importantly we would expect faster delivery from TOTSCo because our customer data will no longer be visible to TOTSCo. This would simplify everything and is ultimately beneficial for the entire industry.	NO. As a CP, we use E2EE encryption when necessary to protect our customers. It seem obvious that this is also how TOTSCO should work. AS a clearing house you have no need to read specific messages and this would introduce security vulnerabilities for no benefit.	NO. We can see no reason at all why E2EE would have any material impact on response times.	AT GO-LIVE. We as a company and you as an industry body must ensure the confidentiality of consumer data and deliver the HUB as fast as possible. E2EE simplifies the HUB's development and processes while protecting the industry from industry-wide data breaches.	ANY. How the system is built to pass messages through a neutral is hub is irrelevant as longs as it works and can scale to a multiple of the anticipated volumes.	

Yes; we encrypt and decrypt other things and do not see issues with implementing E2EE.	Yes it would delay development – it adds complexity to the design, adds new on-boarding and operational processes and requires more testing. In terms of testing and going live, it complicates testing, on-boarding and probably adds extra gates to go-live; we expect issues are more likely both in testing and live so decreases confidence, but would expect this could be mitigated by additional testing effort pre-go-live and live testing after a soft launch to prove everything before everyone goes fully live.	Yes, in several ways: • Public / private key management, including key rotation and security • Ensuring keys are available for systems in a secure manner • Receiving messages that we can't decrypt for some reason • Having messages rejected due to recipient being unable to decrypt for some reason • More compute required (probably negligible per message but potentially not so depending on volume)	Possibly, but hard to say to what degree without deeper analysis and design. This summarises the key points of adding E2EE: • Additional possible failure points • Possible extra latency • More complex design	We have a preference for E2EE at go-live but we could accept not having it if we have assurances around the following concerns: • Data sharing agreements in place between us and TOTSCo / Tech Mahindra - including agreements with any sub-processors • Understanding where data processing (note: not storage) is taking place, and is this outside the UK - including whether hub data can be downloaded or accessed in bulk from outside the UK • Understanding of data retention policy, including retention period, any anonymization processes and destruction of data processes - including how we are informed of any changes • Governance around when sub-processors change - including how we are informed of such changes • Technical and organisational measures, including training, around protection of customer data, integrity and confidentiality - including any penalties for failure to protect data • We'd also need assurances around how TOTSCo handles data sharing with third party integrators to the hub i.e. where a message from [Us] to another CP routes as [Us] -> Hub -> TPI -> Other CP. In such cases, the Managed Access Provider is now processing our customer data Note that if E2EE is not being used, we would need assurances for all these	C#	

				concerns before we could start sharing any data (i.e. using the hub), including in any test environment. If E2EE is not there at go-live we don't see adding it later as worthwhile.		

				Yes		
						Don't respond directly to questions, but generally in favour of encryption from go live
We think that CPs that do not have the capability to encrypt PII which they send to third parties are unlikely to meet the UK data security regulations and would be subject to sanction if their PII data was compromised. CPs that do not have this capability should seek to use a third party portal service to interact with the Hub and that third party service provider should ensure that their PII data is properly secured. The TOTSCo Developer's Guides are simple to follow and can be used by CPs to put in place the necessary security measures.	No - We do not anticipate any delay. On the contrary we would expect a faster delivery from TOTSCo because TOTSCo will no longer be a Data Processor and so its operating and security processes can be more simple . Implementing E2EE will help CPs to meet existing and new security regulations and avoid the PII data leaks experienced by the industry in the past.	No – The opposite as our relationship with TOTSCo will be simplified by the fact that TOTSCo will not be acting as a data processor of our PII. We do anticipate complications in not having E2EE and message signing in place as it would make it difficult for us to defend any PII data breach with the relevant UK authorities.	No – adding encryption to messages take fractions of a second and will not be a significant factor compared to the time required to carry out a data query which in most cases we anticipate will be sub-second in any case. Some members have tested adding E2EE to the specified TOTSCo messages and have seen no measurable impact on response times.	It is essential that E2EE is implemented for go live. The defence that it was too difficult for some CPs to implement in the light of a data breach is unlikely to be effective given the guidance from the UK authorities. We must ensure the confidentiality of consumer data and deliver he HUB as fast as possible. E2EE simplify the HUB development and processes while protecting the industry from what has happened in previous data breaches.		

						From a principle level, end-to-end encryption is more favourable to [us]…however, implementing E2EE on the current TOTSCO design will increase complexity for all CPs and therefore would not be supported by [us] (with this specific design).
Yes, [our] system partner has confirmed, based on the draft E2EE specification v0.2, with a common RSA/AES process, they can implement E2EE.	Yes It would prolong the development time and extend the overall timeline, but the full impact can't be predicted yet, but it will certainly increase the risk of not meeting the timeline. It will likely make it more difficult to do initial (manual) testing, as requests/responses can't be crafted/read in plain text as sending/receiving messages to/from our own implementation, prior to connecting to the Hub, can't be made without tooling including E2EE-support.	Not necessarily, but encryption adds an additional level of complexity, which can make it more difficult to troubleshoot when things go wrong, once live. In cases of communication/message exchange failures, the cryptographic process is another possible source-of-failure which must be considered and handled, e.g. incorrect/invalid/missing keys etc.	Not in normal messages exchanges where everything works as anticipated. The cryptographic processes seem simple enough not to affect the processing of a single message in terms of time. In cases where the ultimate recipient may be missing the key used by the sender (for signing), an on-demand request must be made to retrieve the key, which will affect the SLA. Hopefully this will be rare exceptions, but this will still count towards the SLA.	Never - We believe that TLS (encryption at transit) should be enough assumed that involved parties trust that the hub is secure. All parties that act as an endpoint must take all necessary measures to secure their data, which E2EE cannot address by itself. However, if implemented, we believe it should be implemented from the start, although this may affect the go-live/timeline for all parties. Introducing E2EE after go-live, adds additional development and upgrade work and potential disruption as the whole industry has to adopt this standard at the same time when it goes live.	Our system partner uses C# (.NET Framework 4.8 and dotnet 6)	

This respondent requested confidentiality						
Yes, this is feasible and technically possible.	Yes, it would require rework of the work already completed to prepare for integration with the hub and could significantly complicate end-to-end testing.	The suggestion that the public keys should be refreshed on a regular basis could complicate matters; given the response times required, the CPs would most certainly cache the public keys, and this could result in messages being sent / decrypted with different versions of the keys. The CPs would also need to ensure there is no delay between them publishing their public keys on the hub and updating their private key in their own system. All of these can be resolved in some way, but they bring complication.	Additional time would be needed to access the public keys, encrypt and decrypt messages and signatures, as well as any retry mechanisms linked to working across different versions of the keys. Without the Hub in place it is too early to say whether this would affect our ability to meet the SLA, but it may do.	Never or, maybe after go-live. Given the current timelines, the delay in delivering the Hub, the lack of legal requirement for E2EE, it would seem ill-advised to deliver this at go-live.	From our side, Python.	Some thought must be given to market consolidation here too. E.g. Processes will be required to manage Keys to deal with scenarios such as when CP A acquires CP B or worse only acquires CP B's customer base. Provided that the Hub does not store or log messages, there are limited concerns of security breaches within TOTSCo. The communications will take place through HTTPS so limited risk of capturing the messages in transit. From an industry point of view, the metadata is probably as sensitive as the personal

						information, although it carries lesser liability with respect to GDPR, so seeking to over-protect the PI is illogical.
This respondent requested confidentiality						
Yes	No	It is hard to say without a more detailed specification, but our test platform supports GRCP to LRCP testing for each RCP member. Any member connecting via API is not expected any operational complications from the use of our CSP service (but they may have their own challenges). Some members may wish to use our newly launched match upload service feature. This will require a number of GDPR compliance procedural and more complex	No	At go live, if this does not delay the launch. In fact the proposal may accelerate the launch for TOTSCo of the OTS Hub.	Python or Java	

		platform changes that have yet to be fully assessed.				
Yes. [Our] view is that E2EE is necessary to satisfy our statutory duties in relation to data protection and the new Telecoms Security Act 2021. We also require assurance that developer security best practice has been followed to create the TOTSCo SDK as per TSA COP M4.02 https://www.ncsc.gov.uk/files/NCSC-Vendor-Security-Assessment.pdf	Our view is that E2EE is required for statutory reasons and is therefore not optional and provided that TOTSCo can meet the dependencies required before the go-live deadline.	Our view is that E2EE is necessary for statutory reasons and is therefore not optional.	Our view is that E2EE is necessary for statutory reasons and is therefore not optional.	At go live	Python /C#s	
	Estimated 3 month impact on development			On balance, therefore, we think the low security risk vs the regulatory risk of further delay supports a launch without E2EE,		